PENGEMBANGAN PROTOTIPE ROBOT MANIPULATOR DAN SISTEM MONITORING BERBASIS DEEP LEARNING

SKRIPSI

Diajukan sebagai salah satu syarat untuk menyelesaikan
pendidikan diploma empat (D-4) Program Studi Sarjana Terapan
Teknik Mekatronika
Jurusan Teknik Mesin
Politeknik Negeri Ujung Pandang

PUTU HERDY KURNIAWAN
444 19 020

PROGRAM STUDI SARJANA TERAPAN TEKNIK MEKATRONIKA
JURUSAN TEKNIK MESIN
POLITEKNIK NEGERI UJUNG PANDANG
MAKASSAR
2023

HALAMAN PENGESAHAN

Skripsi dengan judul "Pengembangan Prototipe Robot Manipulator dan Sistem Monitoring berbasis *Deep Learning*" oleh Putu Herdy Kurniawan NIM 444 19 020 dinyatakan layak untuk diujikan.

Makassar, 20 Juli 2023

Pembimbing I,	Pembimbing II,
Ir. Remigius Tandioga, M.Eng.Sc.	Dr. Ir. Simon Ka'ka, M.T
NIP. 19621210 199003 1 005	NIP. 19590913 198803 1 001

Mengetahui

Ketua Program Studi,

Dr.Eng. Akhmad Taufik, S.T., M.T.
NIP. 19760413 200812 1 003

HALAMAN PENERIMAAN

Pada hari ini, hari Rabu tanggal 2 Agustus 2023, tim penguji ujian sidang skripsi telah menerima skripsi mahasiswa: Putu Herdy Kurniawan NIM 444 19 020 dengan judul "Pengembangan Prototipe Robot Manipulator dan Sistem Monitoring Berbasis *Deep Learning*".

Makassar, 2 Agustus 2023

Tim Penguji Ujian Sidang Skripsi:

1.	Dr.Eng. Abdul Kadir Muhammad, S.T., PG.Dipl., M.Eng.	Ketua	(.....)
2.	Mukhtar, S.Pd. M.Eng.	Sekretaris	(.....)
3.	Firman Hamzah, S.T., M.T.	Anggota	(.....)
4.	Prof. A.M. Shiddiq Yunus, S.T., M.Eng.Sc., Ph.D.	Anggota	(.....)
5.	Dr. Ir. Simon Ka'ka, M.T.	Anggota	(.....)
6.	Ir. Remigius Tandioga, M.Eng.Sc.	Anggota	(.....)

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa, karena berkat karunia-Nya, penulisan skripsi ini yang berjudul "Pengembangan Prototipe Robot Manipulator dan Sistem Monitoring berbasis *Deep Learning*" dapat diselesaikan dengan baik.

Dalam penulisan skripsi ini tidak sedikit hambatan yang penulis alami. Namun, berkat bantuan berbagai pihak terutama pembimbing, hambatan tersebut dapat teratasi. Sehubungan dengan itu, pada kesempatan dan melalui lembaran ini penulis menyampaikan terima kasih dan penghargaan kepada:

1. Direktur Politeknik Negeri Ujung Pandang, Bapak Ir. Ilyas Mansur, M.T.;
2. Ketua Jurusan Teknik Mesin, Bapak Dr. Ir. Syaharuddin Rasyid, M.T.;
3. Ketua Program Studi Sarjana Terapan Teknik Mekatronika, Bapak Dr.Eng. Akhmad Taufik, S.T., M.T.;
4. Bapak Ir. Remigius Tandioga, M.Eng.Sc. sebagai pembimbing I dan Bapak Dr. Ir. Simon Ka'ka, M.T. sebagai pembimbing II yang telah mencurahkan perhatian dan kesempatannya untuk mengarahkan penulis dalam menyelesaikan skripsi ini;
5. Ibu Sitti Sahriana, S.S., M.App.Ling. yang telah banyak memberikan waktu membimbing penulis hingga meraih beasiswa IISMAVO 2022.
6. Bapak Prof. Hsueh-Ting Chu (朱學亭) yang telah banyak membimbing penulis selama studi di Asia University, Taiwan.

Ucapan terima kasih tak terhingga juga penulis sampaikan kepada orang tua penulis, Ketut Suciati dan Gede Suparta yang selalu mendukung penulis hingga skripsi ini dapat selesai. Kepada saudari kandung penulis, Made Widya yang telah banyak memberikan perhatiannya kepada penulis. Rasa syukur juga penulis sampaikan berkat kehadiran teman-teman penulis yang selalu menemani penulis, terima kasih Adek Feby, juga kepada tiga sahabat penulis, Faris, Alif, dan Luthfi. Terima kasih juga kepada Muh. Ilham yang selalu menjadi sahabat baik penulis selama menempuh studi di Politeknik Negeri Ujung Pandang.

Penulis menyadari bahwa skripsi ini belum sempurna. Oleh karena itu, penulis mengharapkan kritikan dan saran yang bersifat membangun demi kesempurnaan skripsi ini dan demi perbaikan pada masa mendatang. Semoga skripsi ini bermanfaat bagi para pembacanya.

Makassar, 17 Juli 2023

Penulis

DAFTAR ISI

hlm.

DAFTAR TABEL

hlm.

DAFTAR GAMBAR

hlm.

DAFTAR SIMBOL, SATUAN, DAN/ATAU SINGKATAN

SIMBOL	SATUAN	KETERANGAN
P	Horse Power [HP]	Daya Motor Listrik
T	Newton meter [Nm]	Torsi
N_s	Rotation per Minute [Rpm]	Kecepatan Motor Listrik
V	Volt [V]	Tegangan Listrik
I	Ampere [A]	Kuat Arus Listrik
t	Second [s]	Waktu
m	gram [g]	Massa
$\theta_1, \theta_2, \theta_3$	Derajat [°]	Sudut Link-1, 2, 3
L_1, L_2, L_3	milimeter [mm]	Panjang Link-1, 2, 3
W	Watt [W]	Daya Listrik
E	Kilowatt-hour [kWh]	Energi Listrik

DAFTAR LAMPIRAN

hlm.

SURAT PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Putu Herdy Kurniawan

NIM : 444 19 20

menyatakan dengan sebenar-benarnya bahwa segala pernyataan dalam skripsi ini yang berjudul "Pengembangan Prototipe Robot Manipulator dan Sistem Monitoring berbasis *Deep Learning*" merupakan gagasan dan hasil karya saya sendiri dengan arahan komisi pembimbing dan belum pernah diajukan dalam bentuk apa pun pada perguruan tinggi dan instansi mana pun.

Semua data dan informasi yang digunakan telah dinyatakan secara jelas dan dapat diperiksa kebenarannya. Sumber informasi yang berasal atau dikutip dari karya penulis lain telah disebutkan dalam naskah dan dicantumkan dalam skripsi ini.

Jika pernyataan saya tersebut di atas tidak benar, saya siap menanggung risiko yang ditetapkan oleh Politeknik Negeri Ujung Pandang.

Makassar, 30 Juli 2023

Putu Herdy Kurniawan

NIM 444 19 020

PENGEMBANGAN PROTOTIPE ROBOT MANIPULATOR DAN SISTEM MONITORING BERBASIS DEEP LEARNING

RINGKASAN

Robot manipulator merupakan robot yang sangat penting digunakan di dunia industri 4.0. Robot ini sangat efisien dan akurat. Namun penggunaannya masih sangat terbatas akibat harganya yang sangat mahal serta kurangnya implementasi fisika kinematika dan dinamika robot manipulator di Indonesia. Hal ini menyebabkan orang Indonesia lebih sering mengimpor produk luar negeri ketimbang memproduksi sendiri alat ini.

Penelitian ini dilakukan untuk menyemai ilmu fisika kinematika dan dinamika robot manipulator serta memberikan gambaran sederhana mengenai teknologi robot manipulator dengan kecerdasan buatan (*Artificial Intelligence*). Penelitian ini juga bertujuan untuk mempercepat perkembangan ilmu robot manipulator di Indonesia dengan harapan teknologi ini dapat diproduksi secara masif di tanah air. Sehubungan dengan itu, penelitian ini diawali dengan transformasi kode C++ menjadi python, perancangan algoritma robot dengan menggunakan *deep learning* dan *machine learning*, serta integrasinya ke dalam robot. Pengumpulan data dilakukan dengan hasil *training* model *deep learning* serta (teknik) pengujian pada servo dynamixel AX-12A, sedangkan analisis data dilakukan dengan regresi linear.

Berdasarkan hasil penelitian dan pembahasan dapat disimpulkan bahwa penggunaan *Artificial Intelligence* (AI) pada robot manipulator dapat mempercepat automasi tugas robot manipulator untuk menentukan titik pengambilan objek pada lokasi atau titik koordinat relatif yang tidak tetap terhadap meja manipulator. Teknologi ini juga sangat akurat dan bisa diandalkan dengan tingkat *error* yang kecil.

DEVELOPMENT FOR PROTOTYPE OF MANIPULATOR ROBOT AND MONITORING SYSTEM BASED ON DEEP LEARNING

SUMMARY

Manipulator robot is a very significant robot used in the industrial world 4.0. This robot is very efficient and accurate. However, its use is still very limited due to the high price and the lack of implementation of the physics of kinematics and dynamics of robot manipulators in Indonesia. This causes Indonesians to import foreign products more often than to produce this tool themselves.

This research was conducted to embed the physics of kinematics and dynamics of robot manipulators and provide a simple description of robot manipulator technology with Artificial Intelligence (AI). This research also aims to accelerate the development of robot manipulator science in Indonesia with the hope that this technology can be massively produced in the country. In this regard, this research begins with transforming C++ code into python, designing robot algorithms using deep learning and machine learning, and integrating them into robots. Data collection was carried out using the results of deep learning model training and (technical) testing on the AX-12A dynamixel servos, while data analysis was carried out using linear regression.

Based on the results of the research and discussion, it can be interpreted that the use of Artificial Intelligence (AI) in the manipulator robot can accelerate the task of the robot manipulator automation to determine the point of taking objects at locations or relative coordinate points that are not fixed to the manipulator table. This technology is also very accurate and reliable with a small error rate.

BAB I PENDAHULUAN

1.1 Latar Belakang

Di era modern saat ini, kemajuan industri mengarah pada otomatisasi produksi. Otomatisasi industri merupakan suatu proses memanfaatkan teknologi dan sistem kontrol untuk mengoptimalkan dan mempercepat proses produksi dan operasi industri. Otomatisasi industri melibatkan penggunaan peralatan mekanik dan elektronik, termasuk robot industri, sistem pemrosesan data, sistem kontrol proses, dan teknologi sensor, untuk mengendalikan dan memantau proses produksi dan operasi. Tujuan dari otomatisasi industri adalah untuk meningkatkan produktivitas, efisiensi, dan kualitas produk serta meminimalkan margin kegagalan dan waktu berhenti (*down-time*) mesin. Otomatisasi industri juga membantu mengurangi biaya dan memperbaiki keselamatan kerja bagi pekerja yang bekerja dengan mesin dan peralatan. Otomatisasi produksi memberi kontribusi terhadap efisiensi dan peningkatan kualitas produk yang dihasilkan.

Salah satu elemen utama dari otomatisasi industri adalah penggunaan sistem robotika yang terdiri dari manipulator mekanis dan sistem kontrol. Kelebihan dari menggunakan robot manipulator yaitu dapat mengurangi biaya produksi. Namun, kelemahannya adalah harga robot manipulator saat ini sangat mahal, namun hal tersebut dapat diatasi dengan jumlah produksi yang meningkat (Goryanina et al., 2018:1).

Robot Manipulator pertama kali ditemukan pada awal tahun 1950-an oleh George Charles Devol, seorang penemu dari Louisville, Kentucky. Dia berhasil membuat penemuan dan mematenkan sebuah manipulator yang dapat diprogram

ulang bernama “Unimate” dari “Universal Automation”. Kemudian selama satu dekade, Devol berusaha untuk menjual produknya di industri tetapi tidak berhasil. Pada tahun 1960-an, seorang pebisnis dan juga insinyur bernama Joseph Frederick Engleberger akhirnya membeli paten robot yang dimiliki oleh Devol. Engleberger berhasil memodifikasinya menjadi sebuah robot industri dan membentuk perusahaan bernama Unimation Inc. untuk memproduksi dan memasarkan robotnya. Karena kesuksesan itu, Engleberger dikenal dalam industri sebagai “Bapak Robotika”.

Dalam robotika, manipulator adalah sebuah alat yang digunakan untuk memanipulasi material/obyek tanpa melakukan kontak fisik secara langsung oleh operator. Awalnya penggunaannya ditujukan untuk menangani bahan radioaktif dan bahan kimia berbahaya, atau untuk digunakan di tempat yang sulit diakses. Saat ini robot manipulator telah digunakan dalam berbagai aplikasi, seperti otomatisasi pengelasan, bedah menggunakan robot, aplikasi luar angkasa, hingga industri manufaktur.

Pada tahun 2021, terjadi peningkatan yang signifikan terhadap volume instalasi robot industri global sehingga menyebabkan rekor baru dalam peningkatan jumlah instalasi robot industri sepanjang sejarah. Hal ini diungkapkan oleh *International Federation of Robotics* (IFR) dalam laporan terbarunya *World Robotics* 2022 (International Federation of Robotics, 2022:9).

Pada Program Studi Sarjana Terapan Teknik Mekatronika, Politeknik Negeri Ujung Pandang. Robot Manipulator telah dibuat oleh peneliti sebelumnya yang bernama Andi Baso dan John Michael Adiputra pada tahun 2017, namun belum

dilengkapi sistem kontrol berbasis *Deep Learning*. Selain itu, robot yang telah dibuat pada penelitian tersebut juga menggunakan PCB yang kurang andal (*not reliable*) dan tidak tahan lama (*not durable*). Hal ini menyebabkan sistem kontrol dan pengembangan menjadi lebih sulit karena harus mengganti PCB yang sudah ada. Perlu adanya modul yang lebih solid dan tahan lama (*durable*) serta mudah dikonfigurasi, sehingga mudah dikembangkan dan mudah dilakukan eksperimen terus menerus oleh peneliti selanjutnya. Maka dari itu perlu adanya pengembangan pada sistem elektronik dan kontrolnya dengan menggunakan *deep learning*, serta meningkatkan sistem monitoring robot manipulator ini sehingga bisa diakses melalui jaringan lokal.

Berdasarkan latar belakang di atas, maka penulis membuat tugas akhir dengan judul "Pengembangan Prototipe Robot Manipulator dan Sistem Monitoring berbasis *Deep Learning*". Prototipe robot manipulator yang dibuat terdiri dari 3 *Degree of Freedom* (DOF) dengan memperkuat sistem kontrol menggunakan *Deep Learning* serta meningkatkan teknologi sistem monitoringnya.

Robot manipulator dan sistem monitoring saat ini menjadi hal yang penting dalam berbagai bidang, termasuk industri, pertanian, dan rumah tangga. Tujuan dari tugas akhir ini adalah untuk mengembangkan sebuah robot manipulator yang efisien dan berkualitas, serta sistem monitoring untuk memantau aktivitas robot tersebut.

Melalui penelitian dan pengembangan ini, diharapkan dapat memberikan solusi bagi masalah yang terjadi dalam pemanfaatan robot manipulator saat ini, sekaligus meningkatkan efisiensi dan akurasi dari sistem monitoring. Hasil akhir

dari tugas akhir ini diharapkan dapat memberikan sumbangsih bagi pengembangan robot manipulator dan sistem monitoring pada masa yang akan datang.

1.2 Rumusan Masalah

Adapun rumusan masalah yang akan dibahas adalah, sebagai berikut:

1. Bagaimana meningkatkan sistem mekanik, elektronik dan kontrol robot manipulator?
2. Bagaimana mengembangkan sistem monitoring robot manipulator?

1.3 Ruang Lingkup Penelitian

Adapun ruang lingkup penelitian dibuat agar dalam pengerjaan proyek akhir ini dapat berjalan dengan baik antara lain sebagai berikut:

1. Pengembangan robot manipulator dalam bentuk prototipe.
2. Pengembangan sistem kontrol robot manipulator 3 DOF.
3. Pengembangan sistem monitoring robot manipulator 3 DOF yang berbasis *Deep Learning*.

1.4 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Meningkatkan sistem elektronik dan kontrol robot manipulator.
2. Mengembangkan sistem monitoring robot manipulator.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut:

1. Dapat memberikan solusi bagi masalah yang terjadi dalam pemanfaatan robot manipulator.

2. Dapat melengkapi media pembelajaran praktikum robotika.
3. Dapat digunakan sebagai media pembelajaran mata kuliah Praktikum Robotika.

BAB II TINJAUAN PUSTAKA

2.1 Pengantar Robotika

Robot adalah perangkat mekanis yang dapat dikendalikan secara otomatis melalui komputer atau pengontrol lainnya untuk melakukan berbagai tugas dan operasi. Robot dapat dibedakan menjadi berbagai jenis, termasuk robot industri, robot mobil, dan robot pribadi.

Robot industri digunakan dalam berbagai industri untuk melakukan tugas seperti pengelasan, pemotongan, dan pembuatan produk. Robot mobil digunakan dalam berbagai aplikasi, termasuk pengangkutan, pemetaan, dan pembersihan. Sedangkan robot pribadi digunakan untuk berbagai keperluan seperti asisten rumah tangga, hiburan, dan pengajaran.

Robot dapat dilengkapi dengan berbagai sensor dan peralatan, seperti kamera, mikrofon, dan alat ukur, yang memungkinkan mereka untuk menerima masukan (*input*) dari lingkungan dan melakukan tugas yang lebih kompleks. Robot juga dapat dikendalikan dengan berbagai metode, termasuk perintah suara, perintah sentuh, dan perintah komputer.

Secara umum, robot digunakan untuk meningkatkan efisiensi, menurunkan biaya, dan meningkatkan kualitas dalam berbagai bidang, mulai dari industri hingga rumah tangga.

2.2 Pengantar Manipulasi dalam Robotika

Manipulasi adalah suatu proses menggunakan tangan untuk mengatur ulang lingkungan seseorang. Manipulasi dapat dikatakan merupakan sebuah seni, teknik,

dan juga sekaligus sebuah disiplin ilmu. Disebut seni karena itu dipraktikkan oleh semua orang, tanpa pemahaman yang sistematis serta mendasar. Disebut disiplin teknik karena ada beberapa alat sistematis untuk menerapkan manipulasi robot ke berbagai masalah. Disebut disiplin ilmu karena merupakan proses yang melibatkan keingintahuan manusia yang dapat dipelajari melalui metode ilmiah (Mason, 2001:1).

Manipulasi lebih dari sekedar hanya mengambil sesuatu dan memindahkannya ke tempat lain. Masalah mengenai "Mengambil dan Menempatkan" merupakan suatu aplikasi yang sangat baik pada manipulasi robot. Robot telah melakukan ini selama beberapa dekade di pabrik-pabrik. Namun dalam praktiknya, tidak ada sistem manipulasi robot yang canggih hingga saat ini yang menggunakan teori kontrol yang ketat untuk merancang atau bahkan memberikan umpan balik tingkat rendah yang menentukan kapan robot membuat dan memecahkan kontak dengan objek yang dimanipulasi, sehingga menyebabkan pergerakan serta dinamika kontrol robot manipulator dapat menjadi lebih halus (Tedrake, 2022:7).

Lengan robot pada umumnya terdiri dari bahu, persendian dan tangan yang bisa berupa sebuah *gripper* atau tangan yang memiliki jari seperti halnya tangan manusia sebagai pengambil objek. Bagian tangan robot dikenal sebagai manipulator tangan, yaitu sistem gerak yang berfungsi untuk manipulasi (memegang, mengambil, mengangkat, memindahkan, mengolah) objek. Untuk melakukan pengambilan objek lengan robot ini dilengkapi dengan *end-effector* (*gripper*) yang berupa jari-jari seperti halnya jari manusia. Lengan robot didesain agar dapat mengikuti gerak sesuai dengan gerakan yang dilakukan oleh gerakan

lengan manusia, *input* pengontrol dibuat dengan potensiometer untuk persendian lengan dan *flex Sensor* yang diletakkan pada jari-jari manusia dengan cara membuat pengendali yang sesuai dengan bentuk lengan dan jari-jari manusia agar dapat digunakan sebagai penggerak sendi-sendi pada lengan robot (Muslimin dkk., 2014:8).

2.3 Konsep Dasar Robot Manipulator

Manipulator merupakan sistem mekanik yang menunjukkan pergerakan dari robot. Sistem mekanik ini terdiri dari susunan *link* (rangka) dan *joint* (engsel) yang mampu menghasilkan gerakan yang terkontrol, sebagai rangkaian umpan balik terbuka maupun rangkaian umpan balik tertutup yang dihubungkan dengan sendi-sendi dan dapat melakukan gerakan-gerakan secara bebas.

Manipulator robot adalah sebuah perangkat atau mesin yang digunakan untuk memindahkan atau menangani objek. Manipulator robot biasanya terdiri dari sekumpulan lengan mekanis yang dapat digerakkan dengan menggunakan motor atau hidrolik. Manipulator robot digunakan dalam berbagai bidang, seperti industri, pembuatan, dan pengiriman.

Derajat kebebasan atau juga sering disebut *Degree of Freedom* (DOF) suatu robot merupakan jumlah gerakan independen yang dapat dilakukan oleh *manipulator* terhadap sistem koordinat yang dapat menyebabkan perubahan posisi atau orientasi manipulator, dalam hal ini dimaksud dengan robot manipulator (Muhammad dkk., 2016: 2).

Robot tangan dengan 3 DOF biasanya memiliki lima sendi, yang memungkinkannya untuk bergerak dalam lima cara yang berbeda. Lima sendi ini

memungkinkan tangan robot untuk bergerak pada sumbu x, y, dan z, serta berputar pada dua sumbu tambahan.

Tiga sendi pertama, yang disebut sendi bahu, siku, dan pergelangan tangan, memungkinkan tangan robot untuk bergerak pada sumbu x, y, dan z secara masing-masing. Sendi-sendi ini memungkinkan tangan robot untuk bergerak ke atas dan ke bawah, ke kiri dan ke kanan, serta ke depan dan ke belakang.

Sendi keempat, yang disebut sendi putar pergelangan tangan, memungkinkan tangan robot untuk berputar di sekitar sumbu z. Sendi ini memungkinkan tangan robot untuk memutar alat atau *gripper* di ujung tangannya.

Sendi kelima, yang disebut sendi pada *gripper*, memungkinkan alat atau *gripper* di ujung tangan robot untuk dibuka dan ditutup. Sendi ini memungkinkan tangan robot untuk memegang dan memanipulasi benda.

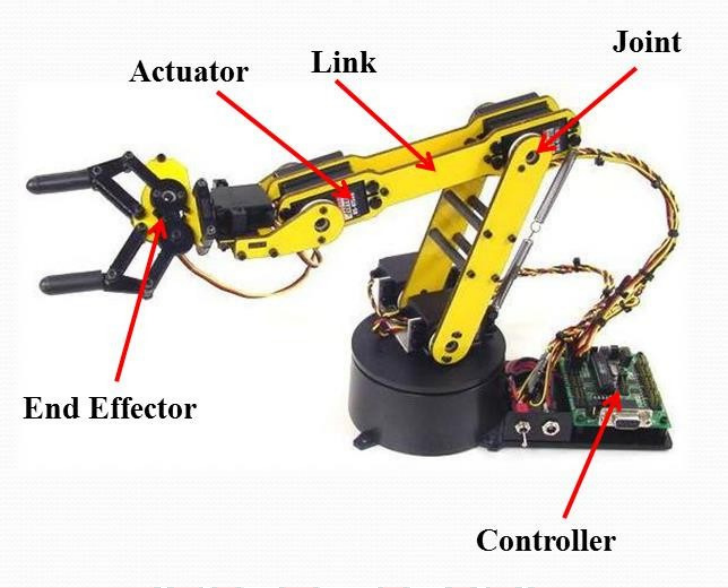
Robot manipulator dengan 3 DOF memungkinkan jangkauan gerak yang lebih luas dan serbaguna, sehingga dapat melakukan berbagai tugas, seperti mengambil dan meletakkan, mengelas, mengecat, dan menyusun bagian-bagian kecil.

2.4 Bagian-Bagian Robot Manipulator

Secara umum manipulator lengan robot terdiri dari:

1. *Joint* (sendi) yaitu koneksi antar *link* yang dapat menentukan pergerakan.
2. *Link* merupakan bagian-bagian kerangka yang kaku yang dihubungkan secara bersamaan sehingga membentuk suatu rangkaian kinematika.
3. *Controller* merupakan bagian dari sistem yang berfungsi untuk mengatur semua kegiatan yang terjadi pada robot.

4. *Actuator* merupakan media penggerak lengan robot. Bisa berupa motor servo, motor DC, dan sebagainya.
5. *End Effector* merupakan aktuator tambahan yang digunakan untuk memegang dan menahan sebuah objek serta dapat digunakan sebagai peralatan-peralatan (*tools*) dalam melakukan operasi tertentu.



Gambar 2.1 Bagian-bagian Robot Manipulator
(Sumber: Baso, 2017)

2.5 Konsep Kinematika Robot Manipulator

Kinematika dalam robotika adalah cabang dari ilmu mekanika yang mempelajari gerak dan posisi suatu objek tanpa memperhatikan penyebab gerak tersebut. Dalam hal ini, objek yang dibahas adalah robot dan bagaimana robot bergerak dan mencapai posisi yang diinginkan.

Ilmu kinematika menggunakan representasi matematis untuk menggambarkan konfigurasi dan gerakan robot. Beberapa konsep yang sering digunakan dalam kinematika termasuk:

1. Transformasi Ruang: Kinematika mempelajari bagaimana mengubah posisi dan orientasi suatu objek dalam ruang. Ini dilakukan dengan menggunakan transformasi seperti translasi dan rotasi.
2. Analisis Jari-jari: Jari-jari adalah garis yang menghubungkan titik tetap pada suatu objek dengan titik yang bergerak pada objek tersebut. Analisis jari-jari mempelajari bagaimana jari-jari terkait dengan gerakan robot.
3. Interpolasi Gerak: Interpolasi adalah teknik untuk memperkirakan posisi dan orientasi suatu objek pada waktu tertentu berdasarkan posisi dan orientasi saat ini dan posisi dan orientasi yang diinginkan.
4. Analisis Denavit-Hartenberg (DH): Analisis ini menggunakan representasi matematis untuk menggambarkan konfigurasi robot, termasuk posisi dan orientasi jari-jari dan bidang pengait.

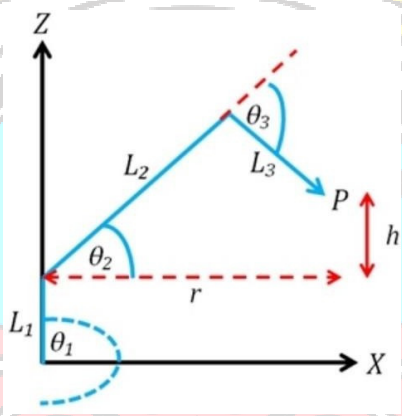
Kinematika memiliki aplikasi penting dalam pemrograman robot dan membantu dalam menentukan bagaimana robot harus bergerak untuk mencapai tujuannya. Misalnya, kinematika dapat membantu dalam memecahkan masalah seperti menentukan jarak dan arah yang harus ditempuh oleh robot untuk mencapai suatu titik tertentu, atau menentukan bagaimana robot harus bergerak untuk melakukan tugas tertentu seperti memegang benda.

Mengenai kinematika dan dinamika pada robot, terdapat penjelasan yang representatif dikemukakan oleh Herizon dan Ade Diana (2014: 67) di bawah ini.

> Kinematika dapat dianalisis dalam dua kajian, yaitu analisa kinematik dan analisa dinamik. Analisa kinematik berkaitan dengan gerakan robot tanpa memandang efek inersia yang terjadi ketika robot melakukan gerakan,

sedangkan analisa dinamik berhubungan dengan efek inersia dari struktur robot secara fisik hasil dari gerakan yang ditimbulkan oleh torsi aktuator ketika robot sedang melakukan pergerakan.

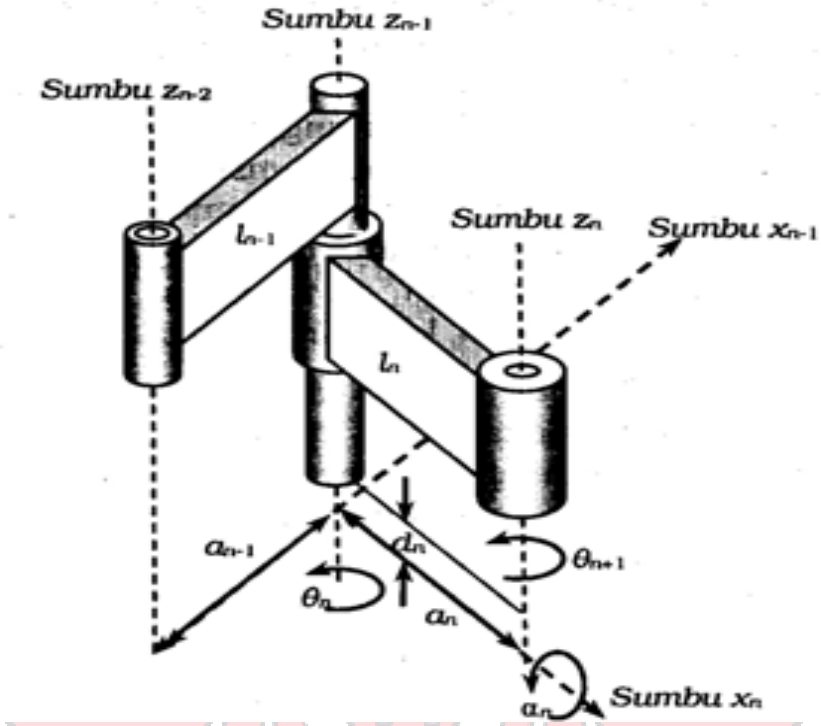
Langkah awal yang dilakukan untuk mendapatkan formulasi robot manipulator 3 DOF yaitu membuat formulasi *inverse kinematic* (kinematikaa mundur) robot manipulator 3 DOF untuk mendapatkan θ_1, θ_2, dan θ_3 (Taufik dkk., 2017:205).



Gambar 2.2 *Inverse Kinematic* 3 DOF
(Sumber: Taufik, 2017)

Model kinematika robot manipulator dapat ditentukan dengan menggunakan metode Denavit-Hertenberg. Prinsip dasar metode ini adalah melakukan transformasi koordinat antar dua *link* yang berdekatan. Hasilnya adalah suatu matriks (4x4) yang menyatakan sistem koordinat dari suatu *link* dengan *link* yang terhubung pada pangkalnya (*link* sebelumnya). Dalam konfigurasi serial, koordinat (ujung) *link*-1 dihitung berdasarkan sendi-0 atau sendi pada tubuh robot. Sistem koordinat *link*-2 dihitung berdasarkan posisi sendi-1 yang berada diujung *link*-1 dengan mengasumsikan *link*-1 adalah basis gerakan *link*-2. Demikian seterusnya, *link*-3 dihitung berdasarkan *link*-2, hingga *link* ke-n dihitung

berdasarkan *link*-(n-1). Dengan cara ini maka tiap langkah perhitungan atau transformasi hanya melibatkan sistem 1-DOF saja. Terakhir, posisi koordinat lengan atau posisi ujung robot/*end-effector* akan dapat diketahui.



Gambar 2.3 Sambungan antar *Link* dan Parameternya
(Sumber: Cahyono, 2015)

Gambar di atas mengilustrasikan dua buah *link* yang terhubung secara seri. Konfigurasi hubungan dapat berupa sendi rotasi ataupun sendi translasi. Dalam hal ini, metode Denavit-Hartenberg (DH) menggunakan 4 buah parameter, yaitu θ, α, *d* dan *a*. Untuk robot n-DOF maka keempat parameter tersebut ditentukan hingga yang ke-n. Penjelasannya yaitu:

- θ_n adalah sudut putaran pada sumbu z_{n-1},
- α_n adalah sudut putaran pada sumbu x_n,

- d_n adalah translasi pada sumbu z_{n-1}, dan
- a_n adalah translasi pada sumbu x_n.

2.6 Matriks Transformasi *Homogeneous*

Berikut bagian-bagian yang penting tentang transformasi *homogeneous*. Matriks rotasi 3x3 yang sebelumnya tidak dapat digunakan untuk menunjukkan pergeseran dari suatu posisi (translasi) dan penskalaan, untuk itu dibutuhkan sebuah matriks baru yang bisa merepresentasikan pergeseran sekaligus penskalaan. Matriks transformasi *homogeneous* merupakan sebuah matriks 4x4. Matriks ini dapat memetakan sebuah vektor posisi yang diekspresikan dalam koordinat *homogeneous* dari suatu sistem koordinat ke sistem koordinat lainnya. Sebuah matriks transformasi *homogeneous* terdiri dari 4 sub-matriks (Fu dkk., 1987).

- Matrik Homogen (4 x 4)

$$T_H = \left[\begin{array}{ccc|c} C1 & -S1 & 0 & x_1 \\ S1 & C1 & 0 & y_1 \\ 0 & 0 & 1 & z_1 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Gambar 2.4 Matriks Homogen (4 x 4)

Sub matriks 3x3 yang terletak di kiri atas merepresentasikan matriks rotasi, sub matriks 3x1 di bagian kanan atas merepresentasikan vektor posisi dari sistem koordinat asal yang dirotasi mengacu pada sistem koordinat referensi. Sub matriks 1x3 di bagian bawah kiri merepresentasikan transformasi perspektif, dan terakhir sub matriks 1x1 yang terletak di bagian kanan bawah adalah matriks yang merepresentasikan faktor penskalaan. Selanjutnya sebuah matriks rotasi 3x3 bisa

diperluas menjadi matriks transformasi *homogeneous* 4x4 yang dilambangkan dengan T_{rot}.

- Matrik Rotasi (3 x 3)

$$T_1 = \begin{bmatrix} C1 & -S1 & 0 \\ S1 & C1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Gambar 2.5 Matriks Rotasi (3 x 3)

2.7 Motor Servo Dynamixel AX-12A

Motor servo Dynamixel AX-12A merupakan aktuator cerdas (*smart actuator*) karena pada masing-masing motor terdapat mikroprosesor yang menyediakan kemampuan komunikasi dengan kontroler, menyediakan informasi tentang posisi dan beban yang bekerja serta temperatur pada motor. Motor servo yang saat ini beredar di pasaran, paling tidak, ada dua varian yakni analog dan digital. Pada Skripsi ini, motor servo yang digunakan adalah motor servo Dynamixel AX-12A. Dynamixel AX-12A merupakan motor servo cerdas yang memiliki torsi hingga 12 kgf.cm dan dilengkapi kemampuan *networking* melalui antarmuka UART TTL *half-duplex multidrop*. Dynamixel AX-12A terdiri dari *reduction gear*, *precision* DC Motor, dan rangkaian kontrol yang telah dilengkapi dengan kemampuan komunikasi (*networking*) dalam satu kemasan. AX-12A merupakan versi terbaru dari AX-12+ dengan kemampuan yang sama tetapi dengan penambahan desain eksternal. Pada gambar 2.2 berikut ini adalah bentuk fisik Dynamixel AX-12A.

Gambar 2.6 Dynamixel AX-12A
(Sumber: Robotis, 2022)

Motor servo ini adalah jenis motor servo digital dengan fitur *daisy-chained*, yang mana satu servo dan yang lainnya, bisa dirangkai secara seri, *daisy-chained*. Masing-masing servo, yang memiliki ID yang unik, diperintah melalui satu jalur serial, dan tidak memerlukan sinyal kontrol yang kontinu (Caysar, 2014: 3).

Berikut ini adalah sajian spesifikasi utama dari servo Dynamixel AX-12A.

Tabel 2.1 Spesifikasi Dynamixel AX-12A

Parameter	**Spesifikasi**
Baud Rate	7.843 [bps] ~ 1 [Mbps]
Berat	54,6 [g]
Resolusi	0.29 [°]
Running Degree	0 ~ 300 [°] Putaran Tak Terbatas
Rasio Roda Gigi	254 : 1
Voltase Masukan	9,0 ~ 12,0 [V] (Rekomendasi: 11,1 [V])
Torsi Maksimum	16,5 [Kgf.cm] atau 1,6 [Nm] (pada tegangan 10 [V])
Torsi *Stall*	1,5 [N.m] (pada tegangan 12 [V]. 1,5 [A]
Koneksi Fisik	TTL Level Multi Drop Bus Half Duplex Asynchronous Serial Communication (8 bit, 1 stop, Tanpa Paritas)
ID	254 ID (0 ~ 253)
Material Roda Gigi	*Engineering Plastic* (*Full*)
Material *Case*	*Engineering Plastic* (Depan, Tengah, Belakang)
Kecepatan Putar	114 [Rpm]

(Sumber: Robotis, 2022)

Dari tabel di atas dapat dihitung daya motor servo sebagai berikut:

$$P = (\frac{T \times N_s}{5252})$$

$$P = (\frac{1.6\ [Nm] \times 114[Rpm]}{5252})$$

$$P = 0{,}03\ HP$$

Di mana:

P = Daya motor [HP]

T = Torsi [Nm]

N_s = Kecepatan motor listrik (Rpm)

5252 = Nilai konstanta (ketetapan) untuk daya motor

2.8 Motor Servo MG996R

Gambar 2.7 Motor Servo MG996R
(Sumber: Components 101, 2019)

Motor servo MG996R adalah jenis motor servo standar yang digunakan dalam robotika dan kontrol sistem. Motor ini merupakan *metal gear* servo yang memiliki rotor DC dan stator dengan lilitan dan magnet dan yang dikendalikan oleh sinyal PWM. MG996R memiliki torsi yang tinggi dan akurasi posisi yang baik, dan bisa

berputar hingga 180 derajat. Motor ini biasanya digunakan dalam aplikasi seperti pengendalian kamera, mekanik, dan kontrol mesin.

Tabel 2.2 Spesifikasi Motor Servo MG996R

Parameter	**Spesifikasi**
Tegangan Operasi	5 [V]
Arus Maksimum	2.5 [A] pada tegangan 6 [V]
Torsi *Stall*	9.4 [Kgf.cm] pada tegangan 4.8 [V]
Torsi *Stall* Maksimum	11 [Kgf. cm] atau 1 [Nm] pada tegangan 6 [V]
Kecepatan Operasi	0.17 [s] / 60° atau 58 [Rpm]
Tipe Roda Gigi	*Metal*
Rotasi	0° – 180°
Berat Motor	55 [g]

(Sumber: Components 101, 2019)

Diasumsikan torsi *stall* maksimum sebagai torsi maksimum motor. Maka dapat dihitung daya motor sebagai berikut:

$$P = (\frac{T \times N_s}{5252})$$

$$P = (\frac{1\ [Nm] \times 58\ [Rpm]}{5252})$$

$$P = 0{,}011\ HP$$

Di mana:

P = Daya motor [HP]

T = Torsi [Nm)

N_s = Kecepatan motor listrik (Rpm)

5252 = Nilai konstanta (ketetapan) untuk daya motor

2.9 Motor Servo ES08MA

Gambar 2.8 Motor Servo ES08MA
(Sumber: EMAX, 2020)

Motor servo ES08MA adalah jenis motor servo analog mini yang sering digunakan dalam aplikasi robotika khususnya *remote control* (RC) pesawat terbang. Motor ini memiliki ukuran kecil, ringan, dan kompak, sehingga cocok digunakan dalam berbagai proyek yang membutuhkan mekanisme gerakan yang akurat dan dapat diatur. ES08MA memiliki torsi yang relatif cukup tinggi untuk ukurannya dan akurasi posisi yang baik, serta dapat berputar hingga 180 derajat.

Tabel 2.3 Spesifikasi Motor Servo ES08MA

Parameter	**Spesifikasi**
Tegangan Operasi	4,8 [V] ~ 6,0 [V]
Torsi *Stall*	1,6 [Kgf.cm] atau 0,15 [Nm] pada tegangan 4,8 [V]
Kecepatan Operasi	0,12 [s] / 60° atau 83 Rpm tanpa beban
Berat Motor	12 [g]
Dimensi	23 x 11,5 x 24 [mm]
Jenis	Analog
Bahan Roda Gigi	Metal

(Sumber: EMAX, 2020)

Berdasarkan data pada tabel di atas dapat dihitung daya motor sebagai berikut:

$$P = (\frac{T \times N_s}{5252})$$

$$P = (\frac{0{,}15\ [Nm] \times 83\ [Rpm]}{5252})$$

$$P = 0{,}002\ HP$$

Di mana:

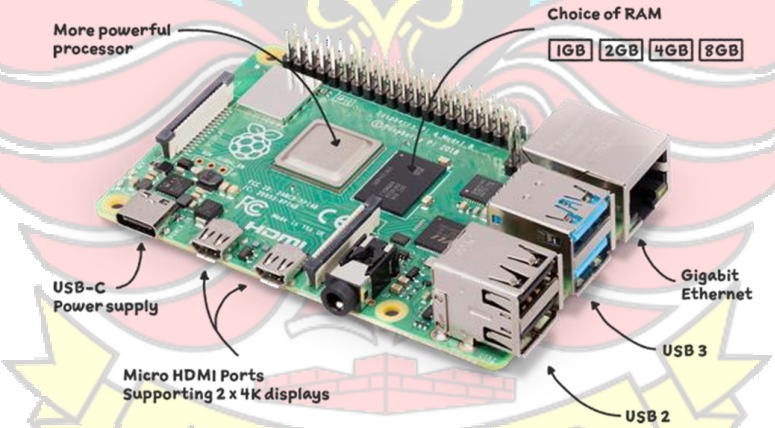
P = Daya motor [HP]

T = Torsi [Nm]

N_s = Kecepatan motor listrik (Rpm)

5252 = Nilai konstanta (ketetapan) untuk daya motor

2.10 Raspberry Pi



Gambar 2.9 Raspberry Pi 4 Model B
(Sumber: Raspberry, 2019)

Raspberry Pi 4 Model B adalah salah satu komputer *single-board* yang dikembangkan oleh Raspberry Pi Foundation. Raspberry Pi 4 Model B dilengkapi dengan fitur-fitur dan spesifikasi yang lebih baik dibandingkan dengan model sebelumnya, membuatnya menjadi pilihan populer bagi pengembang, pencinta

teknologi, dan pengguna umum yang ingin membangun proyek DIY atau menggunakan komputer kecil yang dapat diprogram.

Berikut ini adalah beberapa spesifikasi dan fitur Raspberry Pi 4 Model B secara detail:

Tabel 2.4 Spesifikasi Raspberry Pi 4 Model B

Parameter	**Spesifikasi**
SoC	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit @ 1.5 GHz
RAM	1 GB, 2 GB, 4 GB, atau 8 GB LPDDR4-3200 SDRAM
Memori	microSD
GPIO	Raspberry Pi standard 40 pin GPIO header
Konektivitas	• 2.4 GHz dan 5.0 GHz IEEE 802.11 ac wireless • Bluetooth 5.0 BLE • Gigabit Ethernet
Multimedia	• H.265 (4kp60 decode) • H264 (1080p60 decode, 1080p30 encode) • OpenGL ES 3.1, Vulkan 1.0
Port	• 2 × USB 3.0; 2 × USB 2.0 • 2 × micro-HDMI (up to 4kp60 fps) • 2-lane MIPI DSI display • 2-lane MIPI CSI camera • 4-pole stereo audio dan composite video
Power	• 5V DC via USB-C connector (up to 3A) • 5V DC via GPIO header (up to 3A) • Power over Ethernet (PoE) enabled (requires separate PoE HAT)
Dimensi	85 mm × 56 mm

Sumber: (Raspberry, 2019)

1. Prosesor dan RAM:

Raspberry Pi 4 Model B dilengkapi dengan prosesor quad-core ARM Cortex-A72 64-bit yang berjalan pada kecepatan 1,5 GHz. Prosesor ini memiliki performa yang jauh lebih baik dibandingkan dengan model sebelumnya. Selain itu, Raspberry Pi 4 Model B juga memiliki opsi RAM yang lebih besar, yaitu 2GB, 4GB, atau 8GB LPDDR4-3200 SDRAM. Opsi RAM yang lebih besar memungkinkan komputer ini untuk menjalankan aplikasi yang lebih besar atau kompleks.

2. Konektivitas:

Raspberry Pi 4 Model B memiliki konektivitas yang lebih baik dibandingkan dengan model sebelumnya. Komputer ini dilengkapi dengan dua port micro-HDMI yang mendukung *output* video 4K hingga 60 fps. Selain itu, Raspberry Pi 4 Model B juga memiliki *port* Gigabit Ethernet, dua *port* USB 3.0, dua *port* USB 2.0, dan *port* USB-C untuk daya. Ada juga *slot* kartu microSD untuk menyimpan sistem operasi dan data.

3. Jaringan nirkabel:

Raspberry Pi 4 Model B memiliki dukungan untuk jaringan nirkabel. Komputer ini dilengkapi dengan Wi-Fi *dual-band* 802.11ac dan Bluetooth 5.0 BLE. Dengan dukungan ini, pengguna dapat menghubungkan Raspberry Pi 4 Model B ke jaringan Wi-Fi dan perangkat Bluetooth.

4. Kompatibilitas:

Raspberry Pi 4 Model B kompatibel dengan banyak sistem operasi yang berbeda, termasuk Raspbian, Ubuntu, dan banyak distribusi Linux lainnya. Selain itu, Raspberry Pi 4 Model B juga mendukung bahasa pemrograman seperti Python,

C++, dan Java, sehingga memungkinkan pengguna untuk membuat berbagai aplikasi dan proyek DIY.

5. Harga:

Harga Raspberry Pi 4 Model B tergantung pada opsi RAM yang dipilih. Harga mulai dari sekitar $35 untuk model dengan RAM 2GB hingga sekitar $75 untuk model dengan RAM 8GB. Harga ini membuat Raspberry Pi 4 Model B menjadi komputer *single-board* yang relatif terjangkau.

Secara keseluruhan, Raspberry Pi 4 Model B menawarkan performa yang lebih baik dan konektivitas yang lebih baik dibandingkan dengan model sebelumnya, membuatnya menjadi pilihan yang baik untuk pengguna yang ingin membangun proyek DIY atau menggunakan komputer kecil yang dapat diprogram.

2.11 USB *Communication Converter* U2D2

Gambar 2.10 USB *Communication Converter* U2D2
(Sumber: Robotis, 2023)

U2D2 merupakan sebuah konverter komunikasi dengan *port* USB yang memungkinkan pengontrolan dan pengoperasian servo Dynamixel melalui PC.

Dengan konverter ini, servo dynamixel dapat dikontrol langsung dengan Raspberry Pi 4.

Berikut adalah tabel spesifikasi *communication converter* ini:

Tabel 2.5 Spesifikasi USB *Communication Converter* U2D2

Parameter	**Spesifikasi**
Berat	9 [g]
Dimensi	48 × 18 × 14,6 [mm]
Port	3 Pin TTL Level 4 Pin RS-485 4 Pin UART
Baudrate	6 Mbps (Maksimum)

(Sumber: Robotis, 2023)

2.12 End Effector

Kemampuan robot juga tergantung pada piranti yang dipasang pada lengan robot. Piranti ini biasanya dikenal dengan *end effector*. *End effector* ada dua jenis yaitu pencengkram (*gripper*) yang digunakan untuk memegang dan menahan obyek, dan peralatan-peralatan (*tools*) yang digunakan untuk melakukan operasi tertentu pada suatu obyek. Contohnya: melakukan proses *packing*, bor, penyemprot cat, gerinda, pengelasan (*welding*) dan lain sebagainya.

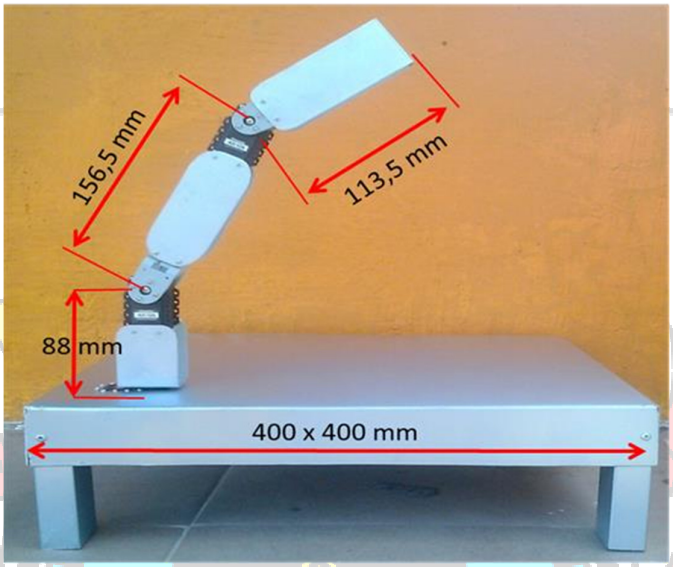
Gambar 2.11 *End Effector* sebagai *Gripper*
(Sumber: Bernier, 2021)

Gambar 2.12 *End Effector* sebagai *Tools (Welding)*
(Sumber: Genesis System, 2023)

2.13 Tinjauan Penelitian Sebelumnya

Penelitian terkait robot manipulator ini telah dilakukan dua kali sebelumnya. Penelitian pertama pada tahun 2016 berjudul "Rancang Bangun Prototipe Robot Manipulator" oleh Jamil Mustarin dan Miansari Mogot. Pada penelitian tersebut, Jamil dan Miansari membuat robot manipulator 3 *Degree of Freedom* (DOF) tanpa *end effector* dengan menggunakan mikrokontroler Arduino Uno, serta memformulasikan matriks rotasi, matriks transformasi homogen dan vektor posisi

robot manipulator untuk kemudian dilakukan pengujian terhadap akurasi dari pergerakan (*movement*) robot manipulator.



Gambar 2.13 Robot Manipulator pada Penelitian Pertama
(Sumber: Mustarin, 2016)

Penelitian tersebut kemudian dikembangkan pada tahun 2017 atau satu tahun setelah penelitian pertama. Penelitian kedua dilakukan oleh Andi Baso dan John Michael Adiputra dengan judul "Pengembangan Prototipe Robot Manipulator". Pada penelitian ini Baso dan Michael menambahkan *end effector* 2 DOF pada konstruksi robot, mengganti mikrokontrolernya menjadi Arduino Mega, memperbaiki panel kontrol dan sistem monitoring, serta menambahkan *emergency button* pada panel robot.

Gambar 2.14 Robot Manipulator pada Penelitian kedua
(Sumber: Baso, 2017)

BAB III METODE PENELITIAN

3.1 Tempat dan Waktu Penelitian

Penelitian ini akan dilakukan di Gedung Pasca Sarjana, kampus Politeknik Negeri Ujung Pandang. Adapun waktu pelaksanaan penelitian dimulai dari bulan Februari 2023 sampai dengan bulan Agustus 2023. Jadwal pelaksanaan dapat dilihat pada tabel 3.1 di bawah ini.

Tabel 3.1 Jadwal Kegiatan Penelitian

No	Kegiatan	Bulan																													
		Januari				Februari				Maret				April				Mei				Juni				Juli				Agustus			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Studi Literatur																																
2	Pembuatan dan Seminar Proposal																																
3	Revisi Proposal																																
4	Pengecekan Mekanik dan Elektronik																																
5	Pengadaan Alat dan Bahan																																
6	Pembuatan Mekanik dan Elektronik																																
7	Pembuatan Program																																
8	Uji Coba Alat																																
9	Seminar Hasil																																
10	Revisi Seminar Hasil																																
11	Pembuatan Paper																																

3.2 Alat dan Bahan

Untuk mendukung pelaksanaan dalam penelitian ini, terdapat beberapa alat dan bahan yang diperlukan agar tugas akhir dapat dikerjakan sesuai rencana, serta memenuhi kriteria dan tujuan yang telah ditetapkan sebelumnya pada Bab I. Adapun alat dan bahan yang akan digunakan dalam pelaksanaan penelitian dapat dilihat pada Tabel 3.2 dan Tabel 3.3 di bawah ini.

Tabel 3.2 Alat yang Digunakan pada Penelitian

No.	Alat
1	Komputer/Laptop
2	Obeng elektronik
3	Kunci L
4	Obeng (+) & (-)
5	Bor Tangan Listrik
6	Gergaji Besi
7	*Power Supply*
8	Solder
9	Multimeter Digital
10	*Cutter*
11	Adaptor
12	Penggaris

Tabel 3.3 Bahan yang Dibutuhkan dalam Penelitian

No	Bahan	Jumlah
1	Raspberry Pi 4 8 Gb RAM	1 unit
2	USB Communication U2D2	1 unit
3	Raspberry Pi Camera	1 unit
4	Motor Servo ES08MA	1 unit
5	Kabel Jumper	Secukupnya
6	*Banana Jack Connector*	2 buah
7	Akrilik	2 set
8	Pembungkus Kabel	1 gulung
9	Tali Sigma	Secukupnya
10	*Wire Cable Carrier*	1 gulung
11	Isolasi listrik	1 gulung
12	Baut	Secukupnya
13	Resistor	Secukupnya
14	PCT Terminal	1 unit
15	Ring Baut	Sesuai kebutuhan
16	Pin Header	Sesuai kebutuhan
17	Kawat Timah	Secukupnya
18	*Spacer*	Sesuai kebutuhan

Selain alat dan bahan yang tertera pada tabel di atas, dalam penelitian ini juga terdapat beberapa *software/platform* yang diperlukan untuk mengerjakan maupun mendukung penelitian. Berikut adalah daftar *software/platform* yang akan digunakan:

- Thonny
- Dynamixel Wizard 2.0
- RealVNC Viewer
- Command Prompt
- Putty.exe
- Xampp Control Panel
- Nmap – Zenmap GUI
- Microsoft Visual Studio Code
- Git / Github (*version control*)
- Autodesk Inventor Professional 2023

3.3 Langkah Kerja

3.3.1 Langkah Kerja

Metode penelitian yang telah digunakan dalam proses pelaksanaan, pembuatan, dan analisis penelitian ini adalah sebagai berikut:

1. Studi Literatur

   Untuk memperoleh landasan teori dalam penelitian dan pembuatan alat, maka tahap pertama yang dilakukan adalah mengumpulkan berbagai informasi yang berkaitan robot manipulator 3 DOF secara umum, atau

yang menggunakan teknologi servo Dynamixel AX-12A dari robotis secara khusus. Adapun referensi yang digunakan adalah buku-buku acuan, jurnal-jurnal, artikel-artikel, serta informasi yang diperoleh dari internet.

2. Formulasi dan Simulasi

   Formulasi dilakukan untuk memodelkan sebuah sistem dalam analisis kinematika, sehingga akan diperoleh model matematisnya. Simulasi dilakukan untuk melihat hasil dari formulasi yang telah dibuat.

3. Eksperimen

   - Mekanik

     Pembuatan perangkat keras dari robot manipulator yang telah dirancang.

   - Elektronik

     Pembuatan rangkaian elektronika untuk *driver* kontroler yang menggerakkan robot manipulator

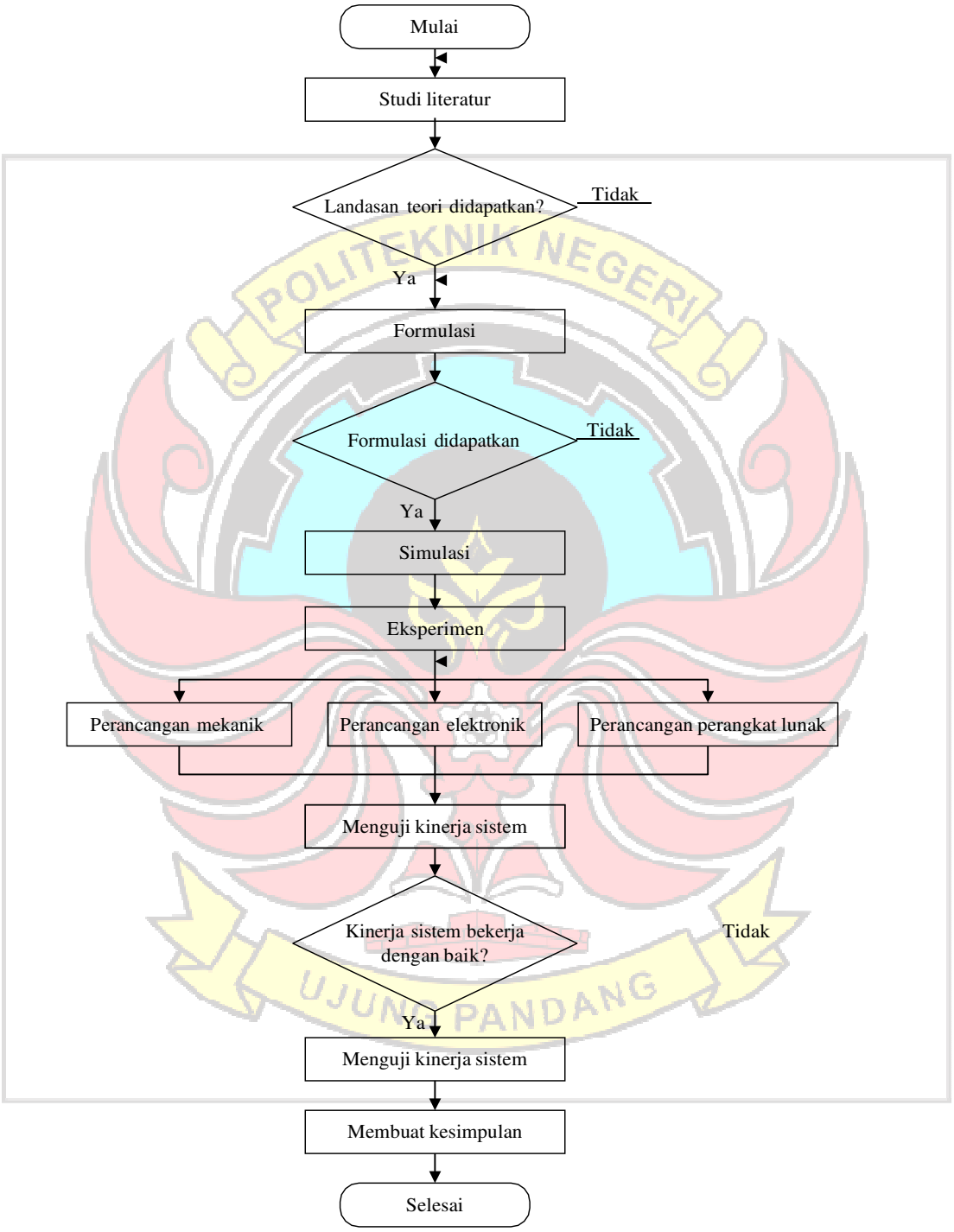
   - Kontrol

     Pembuatan perangkat lunak (program) pada robot manipulator.

4. Menguji kinerja sistem secara keseluruhan serta mengambil data dari hasil pengujian yang dibuktikan perhitungan matematis.

5. Menganalisis hasil serta menarik kesimpulan.

3.3.2 Diagram Alir Penelitian

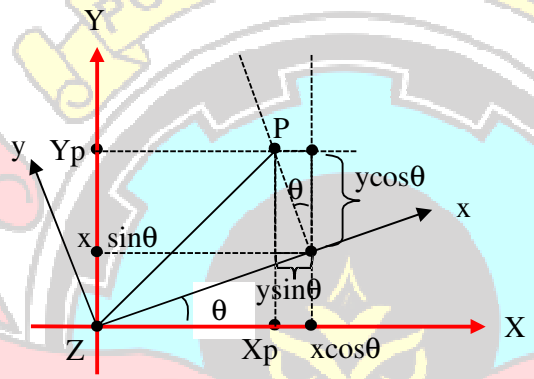


Gambar 3.1 Diagram Alir Penelitian

3.3.3 Formulasi Robot Manipulator

Untuk mendapatkan formulasi manipulator 3 DOF ada beberapa langkah yang perlu diperhatikan seperti rotasi matriks terhadap sumbu koordinat bumi X, Y, dan Z. Pada formulasi ini juga diuraikan vektor posisi robot manipulator 3 DOF sehingga dapat diketahui posisi ujung link manipulator (titik P).

a. Matriks rotasi mengelilingi sumbu Z



Gambar 3.2 Matriks Rotasi Mengelilingi Sumbu Z

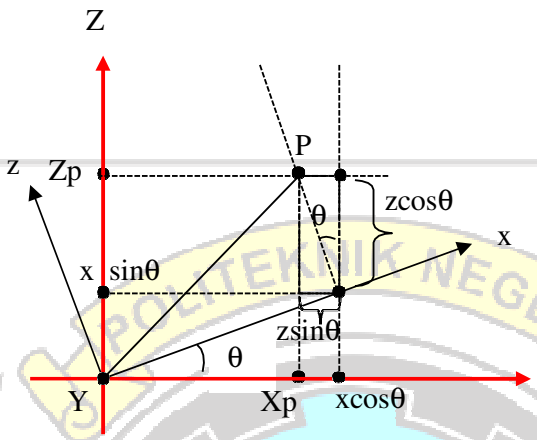
Dari gambar di atas maka akan didapat persamaan berikut ini:

$$Xp = x\cos\theta - y\sin\theta$$
$$Yp = x\sin\theta + y\cos\theta$$
$$Zp = z$$

$$\begin{Bmatrix} Xp \\ Yp \\ Zp \end{Bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$$

$$Rz = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

b. Matriks rotasi mengelilingi sumbu Y



Gambar 3.3 Matriks Rotasi Mengelilingi Sumbu Y

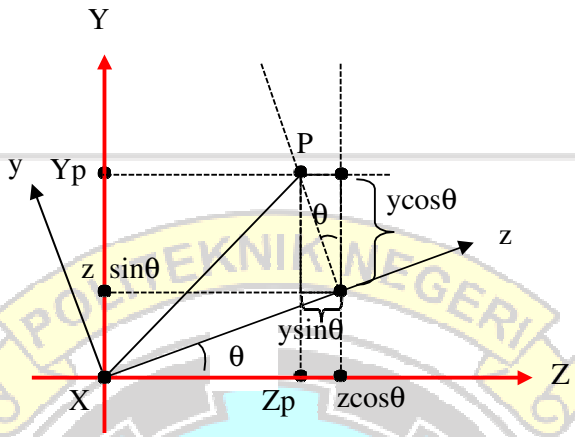
Dari gambar di atas maka akan didapat persamaan sebagai berikut:

$$\begin{aligned} Xp &= x\cos\theta - z\sin\theta \\ Yp &= y \\ Zp &= x\sin\theta + z\cos\theta \end{aligned}$$

$$\begin{Bmatrix} Xp \\ Yp \\ Zp \end{Bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$$

$$Ry = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

c. Matriks rotasi mengelilingi sumbu X



Gambar 3.4 Matriks Rotasi Mengelilingi Sumbu X

Dari gambar di atas maka akan didapat persamaan sebagai berikut:

$$\begin{aligned} Xp &= x \\ Yp &= y\cos\theta + z\sin\theta \\ Zp &= z\cos\theta - y\sin\theta \end{aligned}$$

$$\begin{bmatrix} Xp \\ \{Yp\} \\ Zp \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} . \{y\} \begin{matrix} x \\ \\ z \end{matrix}$$

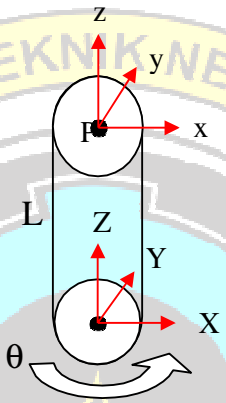
$$Rx = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

d. Vektor Posisi 1 DOF

Pada formulasi robot manipulator, digunakan matriks transformasi homogen yang merupakan kombinasi dari matriks rotasi dan matriks peralihan (translasi). Secara umum matriks transformasi homogen dinyatakan sebagai:

$$T = \begin{bmatrix} R_{3\times3} & P_{3\times1} \\ f_{1\times3} & 1\times1 \end{bmatrix} = \begin{bmatrix} matriks\ rotasi & vektor\ posisi \\ transformasi\ perspektif & penskala \end{bmatrix}$$

Vektor posisi pada robot manipulator 1 DOF bisa didapat dengan menggunakan matriks transformasi homogen. Berikut ini adalah gambar 1 DOF robot manipulator.



Gambar 3.5 1 DOF Manipulator

Dari gambar di atas maka akan didapat koordinat titik P terhadap koordinat titik bumi (X,Y,Z) pada manipulator 1 DOF dengan menggunakan matriks transformasi homogen sebagai berikut:

$$\begin{Bmatrix} Xp \\ Yp \\ Zp \\ 1 \end{Bmatrix} = [Tz] \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix} \quad \text{..... (1)}$$

Di mana nilai:

$$[Tz] = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

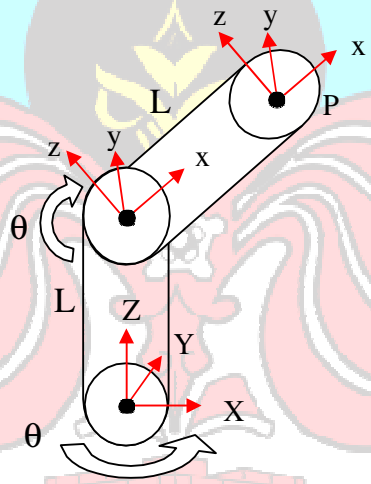
Sehingga:

$$\begin{Bmatrix} Xp \\ Yp \\ Zp \\ 1 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ L_1 \\ 1 \end{Bmatrix} \quad \text{.....(2)}$$

Dari persamaan (2) di atas, maka didapat posisi P terhadap koordinat titik bumi yaitu: X= 0, Y= 0, dan Z= L_1.

e. Vektor Posisi 2 DOF

Vektor posisi pada robot manipulator 2 DOF bisa didapat dengan menggunakan matriks transformasi homogen. Berikut ini adalah gambar 2 DOF prototipe robot manipulator.



Gambar 3.6 2 DOF Manipulator

Dari gambar di atas maka akan didapat koordinat titik P terhadap koordinat titik bumi (X, Y, Z) pada manipulator 2 DOF dengan menggunakan matriks transformasi homogen sebagai berikut:

$$\begin{Bmatrix} Xp \\ Yp \\ Zp \\ 1 \end{Bmatrix} = [\ ][\ ]_{Ty.Tz} \cdot \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix} \quad \text{.....(3)}$$

Di mana nilai:

$$[Ty] = \begin{bmatrix} \cos\theta_1 & 0 & -\sin\theta_1 & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_1 & 0 & \cos\theta_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[Tz] = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Sehingga:

$$\begin{Bmatrix} Xp \\ Yp \\ Zp \\ 1 \end{Bmatrix} = \begin{Bmatrix} L_2 \cos\theta_1 \cos\theta_2 - L_1 \sin\theta_1 \\ L_2 \sin\theta_2 \\ L_2 \sin\theta_1 \cos\theta_2 + L_1 \cos\theta_1 \\ 1 \end{Bmatrix} \quad \text{.....(4)}$$

Dari persamaan (4) di atas, maka didapat posisi P terhadap koordinat titik bumi yaitu:

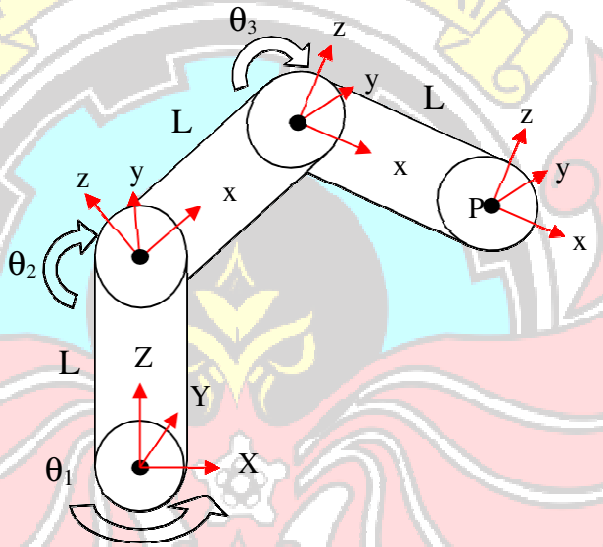
$X = L_2 \cos\theta_1 \cos\theta_2 - L_1 \sin\theta_1$

$Y = L_2 \sin\theta_2$

$Z = L_2 \sin\theta_1 \cos\theta_2 + L_1 \cos\theta_1$

f. Vektor Posisi 3 DOF

Robot manipulator 3 DOF ini sesuai dengan desain robot manipulator yang dirancang. Vektor posisi pada robot manipulator 3 DOF bisa didapat dengan menggunakan matriks transformasi homogen. Berikut ini adalah gambar prototipe robot manipulator 3 DOF.



Gambar 3.7 3 DOF Manipulator

Pada gambar manipulator di atas, dapat diketahui koordinat titik P dengan cara mengamati sumbu koordinat masing-masing *link* (L_1, L_2, L_3) terhadap koordinat titik bumi (X,Y,Z) dengan menggunakan matriks transformasi homogen.

$$\begin{Bmatrix} Xp \\ Yp \\ Zp \\ 1 \end{Bmatrix} = [Tz].[Ty_1].[Ty_2].\begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

Di mana nilai:

$$[Ty_2]=\begin{bmatrix}\cos\theta_3 & 0 & -\sin\theta_3 & L_2\\ 0 & 1 & 0 & 0\\ \sin\theta_3 & 0 & \cos\theta_3 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$[Ty_1]=\begin{bmatrix}\cos\theta_2 & 0 & -\sin\theta_2 & 0\\ 0 & 1 & 0 & 0\\ \sin\theta_2 & 0 & \cos\theta_2 & L_1\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$[Tz]=\begin{bmatrix}\cos\theta_1 & -\sin\theta_1 & 0 & 0\\ \sin\theta_1 & \cos\theta_1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Sehingga

$$\begin{Bmatrix}Xp\\ Yp\\ Zp\\ 1\end{Bmatrix}=\begin{Bmatrix}L_3\cos\theta_1\cos\theta_2\cos\theta_3+L_2\cos\theta_1\cos\theta_2-L_3\cos\theta_1\sin\theta_2\sin\theta_3\\ L_3\sin\theta_1\cos\theta_2\cos\theta_3+L_2\sin\theta_1\cos\theta_2-L_3\sin\theta_1\sin\theta_2\sin\theta_3\\ L_3\sin\theta_2\cos\theta_3+L_2\sin\theta_2+L_3\cos\theta_2\sin\theta_3+L_1\\ 1\end{Bmatrix}\quad\text{.....(6)}$$

Dari persamaan (6) di atas, maka didapat posisi P terhadap koordinat titik bumi yaitu:

$$X = L_3\cos\theta_1\cos\theta_2\cos\theta_3 + L_2\cos\theta_1\cos\theta_2 - L_3\cos\theta_1\sin\theta_2\sin\theta_3$$

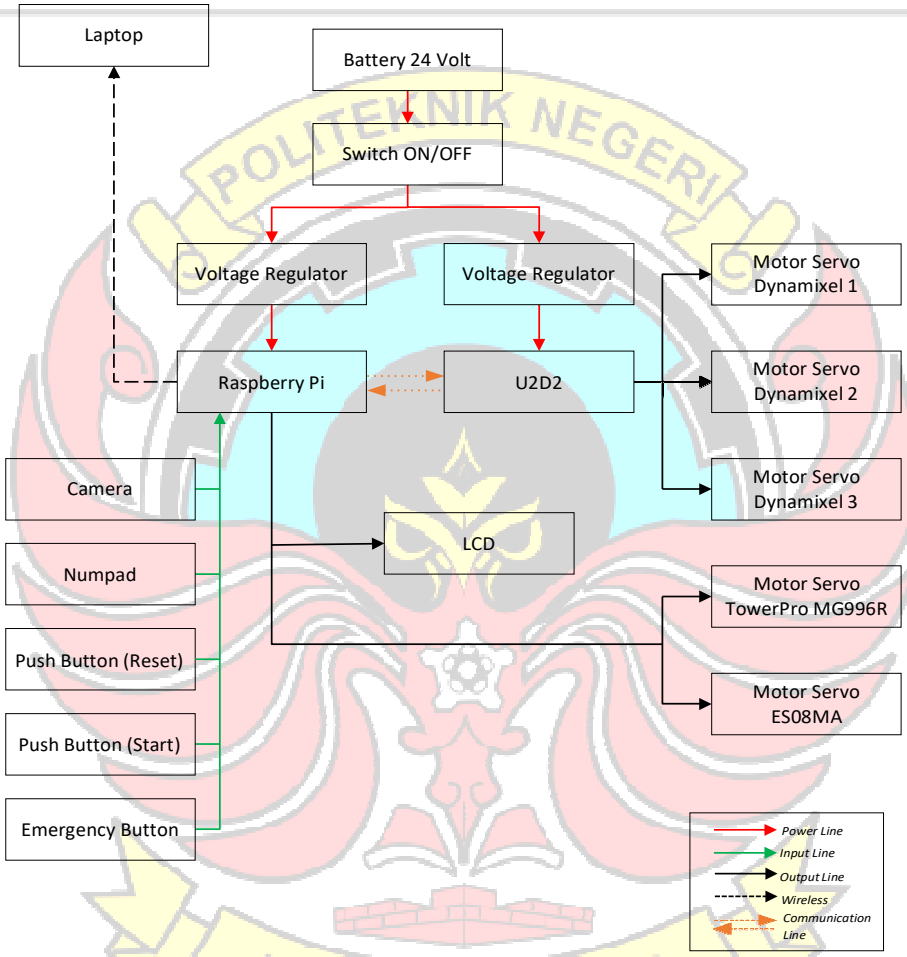
$$Y = L_3\sin\theta_1\cos\theta_2\cos\theta_3 + L_2\sin\theta_1\cos\theta_2 - L_3\sin\theta_1\sin\theta_2\sin\theta_3$$

$$Z = L_3\sin\theta_2\cos\theta_3 + L_2\sin\theta_2 + L_3\cos\theta_2\sin\theta_3 + L_1$$

Formulasi di atas yang akan digunakan untuk mendapatkan koordinat P pada ujung robot manipulator 3 DOF.

3.3.4 Diagram Skematis Alat

Selain diagram alir penelitian, terdapat juga hubungan antar komponen yang digambarkan pada diagram skematis alat berikut:



Gambar 3.8 Diagram Skematis Alat

3.3.5 Parameter (Data) Monitoring

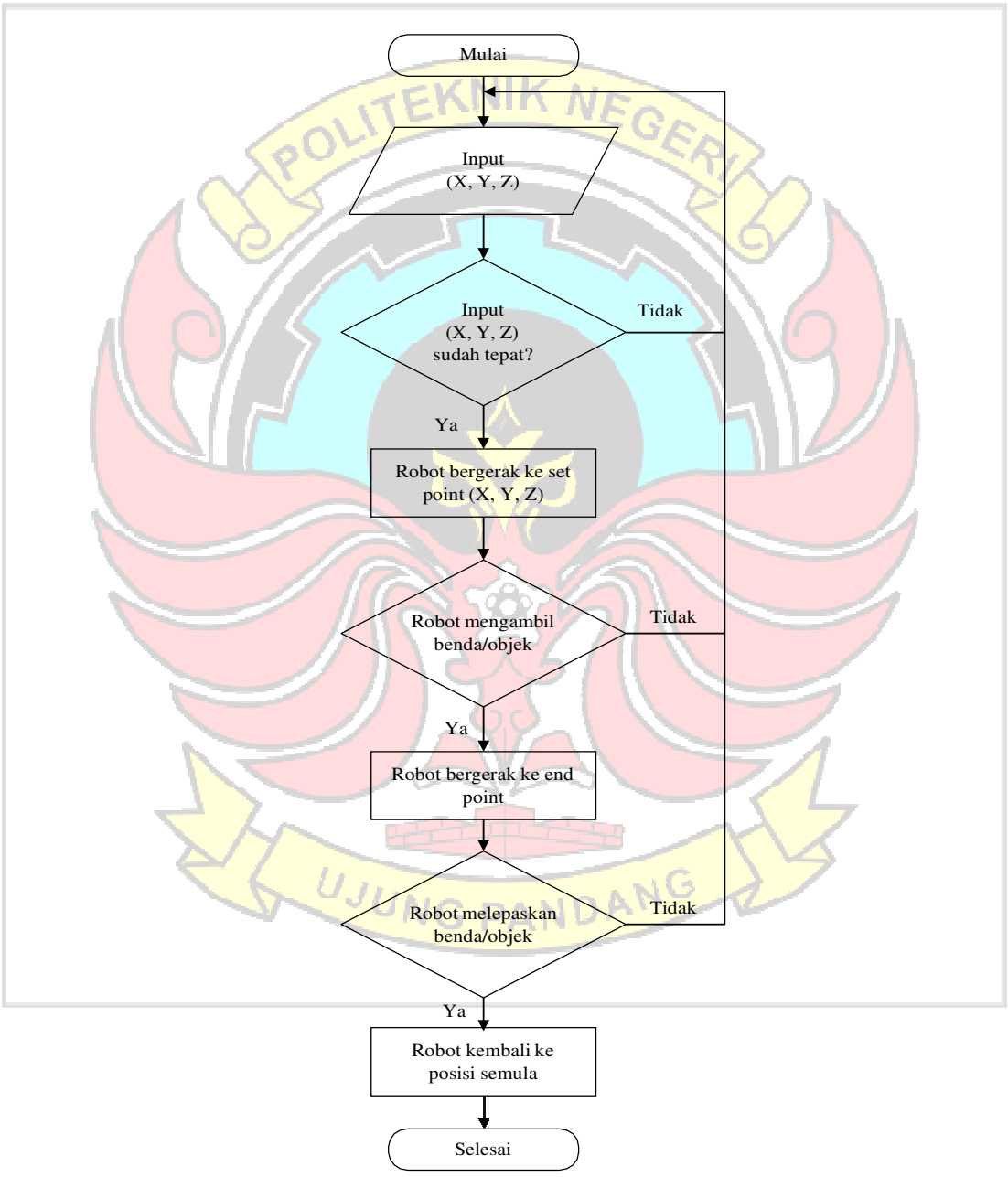
Terdapat 2 data yang akan diambil untuk dimonitoring, yaitu:

1. Posisi servo Dynamixel AX-12A
2. Temperatur motor servo

3.4 Langkah-langkah Pengujian Alat

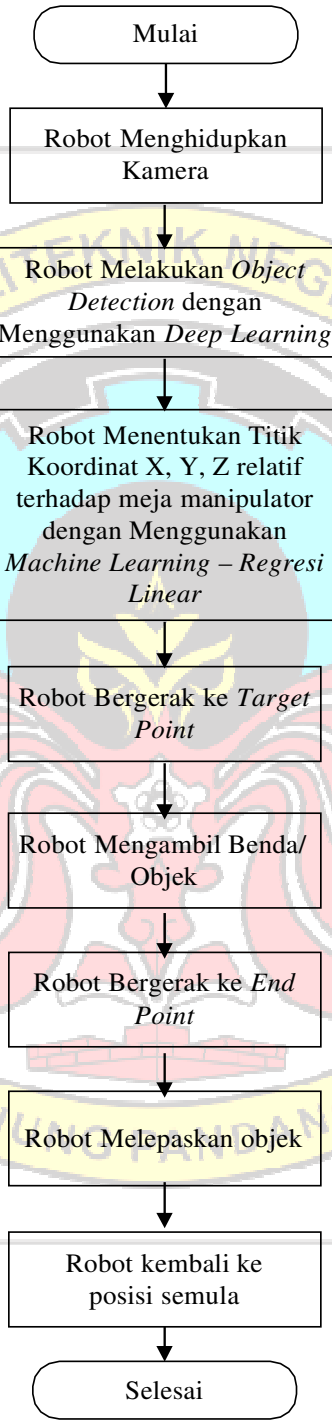
Di bawah ini merupakan diagram alir percobaan dan diagram blok pengontrolan yang akan digunakan pada robot manipulator:

a) Manual



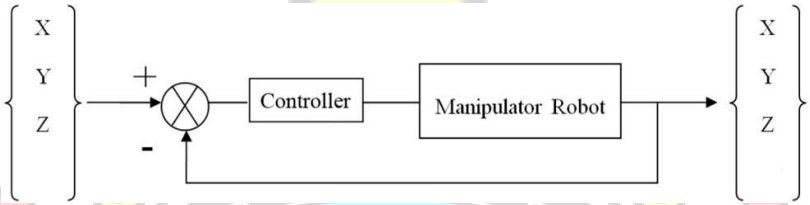
Gambar 3.9 Diagram Alir Percobaan Robot (Manual)

b) Otomatis dengan menggunakan *Deep Learning*



Gambar 3.10 Diagram Alir Percobaan Robot (Auto)

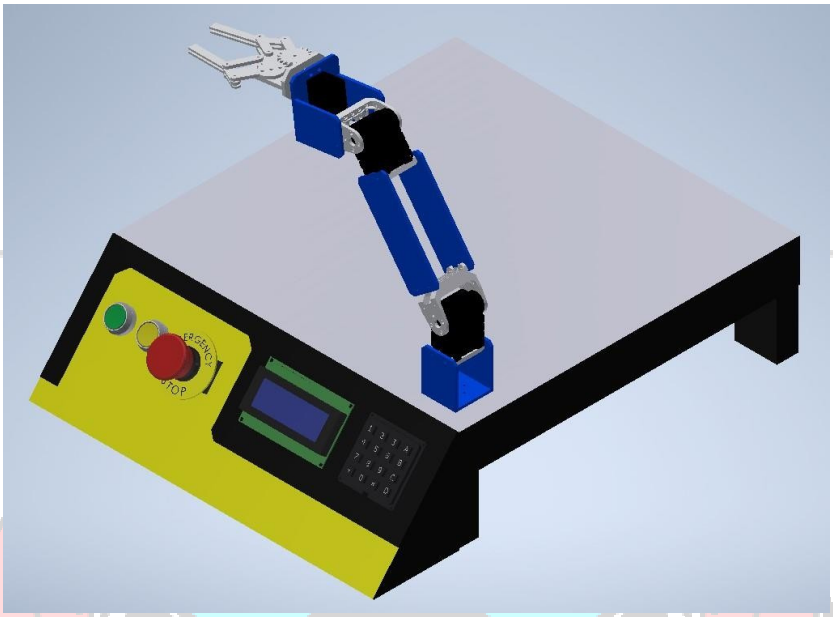
Berikut merupakan gambar diagram blok pengontrolan pada robot manipulator:



Gambar 3.11 Diagram Blok Pengontrolan

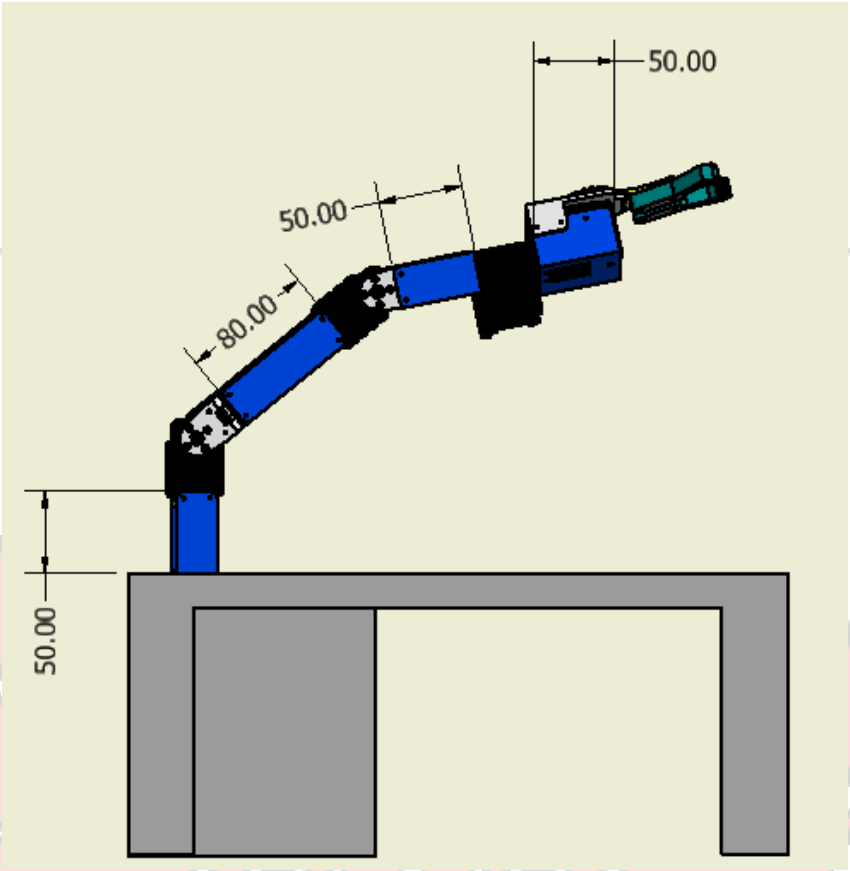
3.5 Teknik Analisis Data

Teknis analisis data yang digunakan yaitu observasi fungsional robot dan sistem monitoring, serta tingkat akurasi titik koordinat keluaran (*output*) terhadap masukan (*input*) yang diberikan. Pengujian fungsional robot manipulator dan sistem monitoringnya bertujuan untuk mengetahui apakah hasil pengembangan yang telah dilakukan dapat berfungsi dengan baik. Jika tidak sesuai harus dilakukan modifikasi sampai menghasilkan unjuk kerja yang baik.



Gambar 3.12 Desain Robot Manipulator

Gambar di atas adalah prototipe robot manipulator yang telah dikembangkan dengan *end effector* dan *emergency button*. Desain tersebut diperbaharui dengan menggunakan *software* Autodesk Inventor 2020 dengan satuan ukuran gambar *metrics* (milimeter). *End effector* pada Robot Manipulator di atas digerakkan oleh motor servo Dynamixel AX-12A.



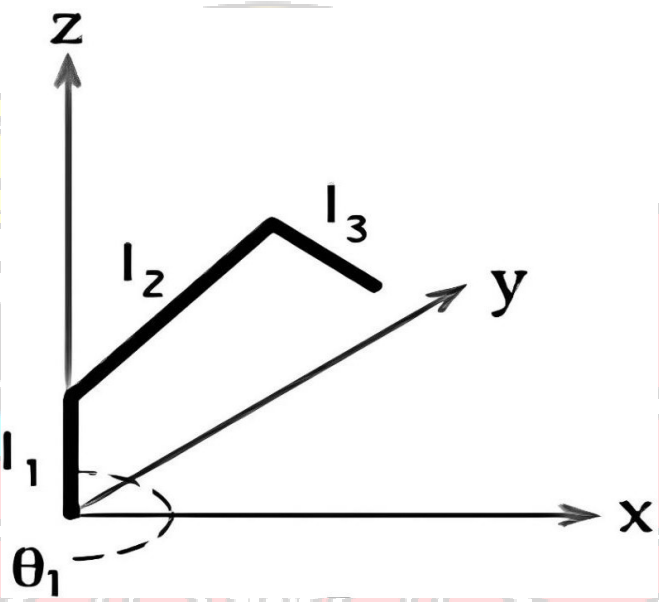
Gambar 3.13 Dimensi Robot Manipulator dalam Satuan Milimeter

BAB IV HASIL DAN PEMBAHASAN

4.1 Hasil

4.1.1 Hasil Pekerjaan Mekanika

4.1.1.1 Formulasi

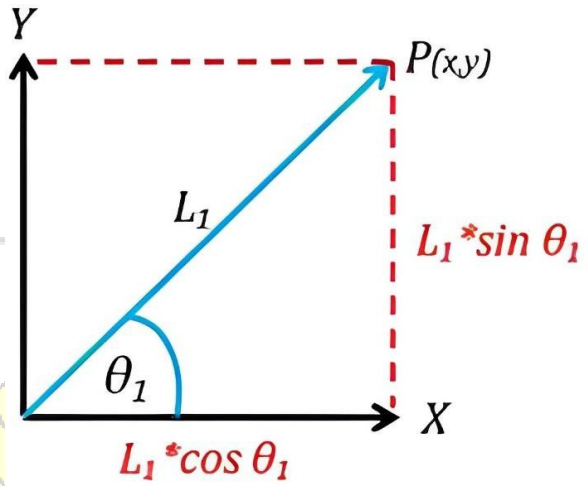


Gambar 4.1 Kinematika 3 DOF

Untuk mendapatkan formulasi *inverse kinematic* (kinematika mundur) robot manipulator 3 DOF, terdapat beberapa langkah yang perlu diperhatikan untuk mendapatkan θ_1, θ_2, dan θ_3, yaitu sebagai berikut:

- Rumus untuk θ_1

  Untuk mendapatkan nilai θ_1 , lihat DOF pertama robot menggunakan perspektif 2D atau 90° terhadap bidang gerak θ_1 , dalam hal ini digambarkan dengan menggunakan perspektif sumbu X dan Z seperti berikut:



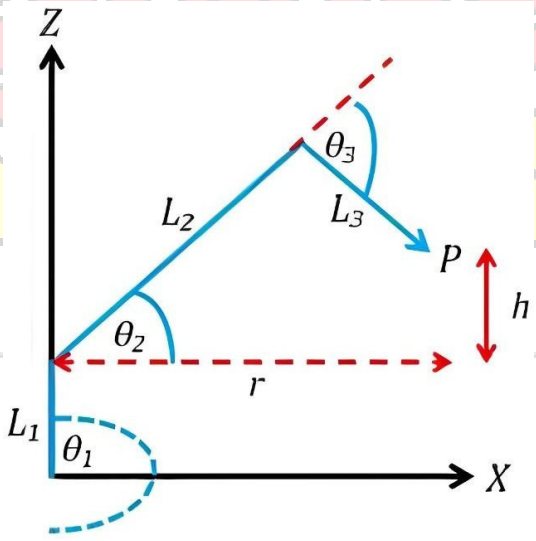
Gambar 4.2 *Inverse Kinematic* untuk θ_1

Dari gambar di atas dapat diperoleh nilai θ_1 sebagai berikut:

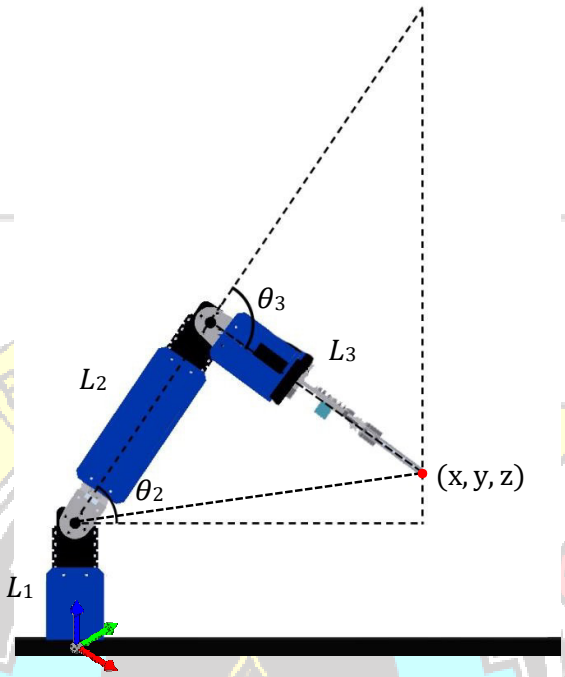
$$\theta_1 = arc\tan\left(\frac{y}{x}\right)$$

- Rumus untuk θ_3

  Sebelum mendapat nilai θ_2 perlu mencari nilai θ_3 terlebih dahulu. Berikut ini adalah langkah untuk mendapatkan nilai θ_3:

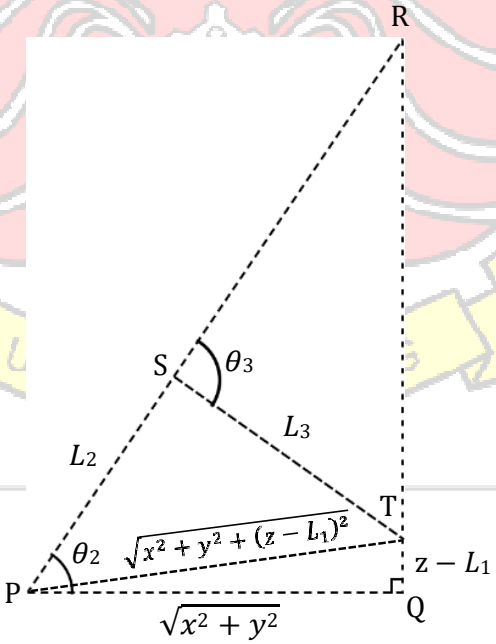


Gambar 4.3 *Inverse Kinematic* untuk θ_3



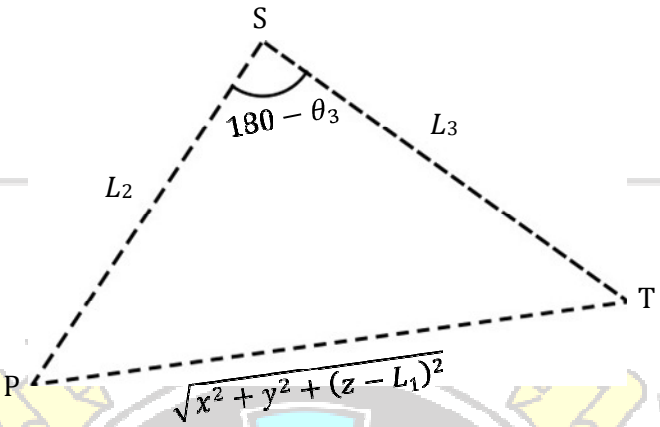
Gambar 4.4 Tampak Samping Robot Manipulator

Mari lihat segitiga siku-siku di atas secara terpisah.



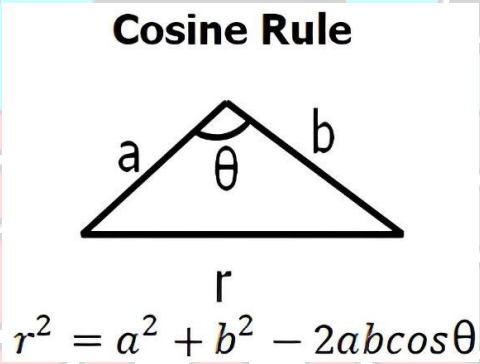
Gambar 4.5 Segitiga pada Observasi Gambar 4.4

Ambil segitiga PST untuk menurunkan nilai θ_3.



Gambar 4.6 Segitiga PST untuk Menurunkan θ_3

Untuk mendapatkan nilai θ_3 gunakan aturan cosinus pada segitiga.



Gambar 4.7 Aturan Cosinus pada Segitiga

Dengan menggunakan aturan cosinus diperoleh rumus seperti berikut:

$$x^2 + y^2 + (z - L_1)^2 = L_2^2 + L_3^2 - 2.L_2.L_3.cos(180 - \theta_3)$$

$$\cos\theta_3 = \frac{x^2 + y^2 + (z - L_1)^2 - L_2^2 - L_3^2}{2.L_2.L_3} \quad \ldots\ldots\ldots\ldots (1)$$

$$\theta_3 = \pm arc\cos\left(\frac{x^2 + y^2 + (z - L_1)^2 - L_2^2 - L_3^2}{2.L_2.L_3}\right)$$

Terdapat 2 konfigurasi robot yang berkaitan dengan penggunaan persamaan di atas, yaitu dengan menggunakan identitas trigonometri:

$$sin^2\,\theta + cos^2\,\theta = 1$$

$$\sin\theta = \sqrt{1 - cos^2\,\theta}$$

1) Untuk *elbow-up*:

$$\sin\theta_3 = +\sqrt{1 - cos^2\,\theta_3} \quad \ldots\ldots\ldots\ldots (2)$$

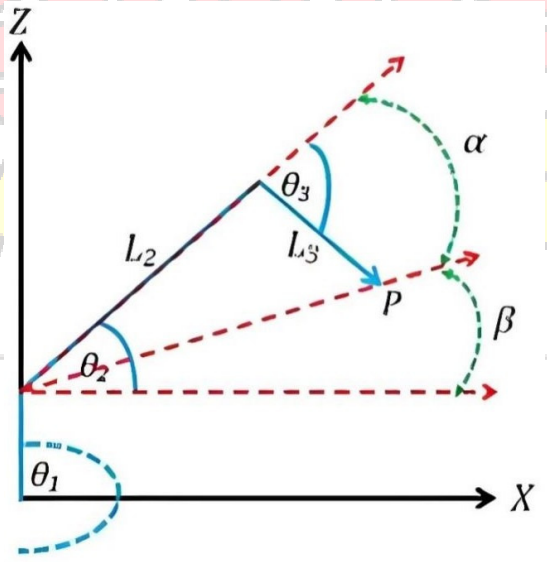
2) Untuk *elbow-down*:

$$\sin\theta_3 = +\sqrt{1 - cos^2\,\theta_3} \quad \ldots\ldots\ldots\ldots (3)$$

Pada robot ini, konfigurasi yang digunakan adalah konfigurasi *elbow-down*. Dari beberapa persamaan di atas dapat diperoleh rumus θ_3 adalah sebagai berikut:

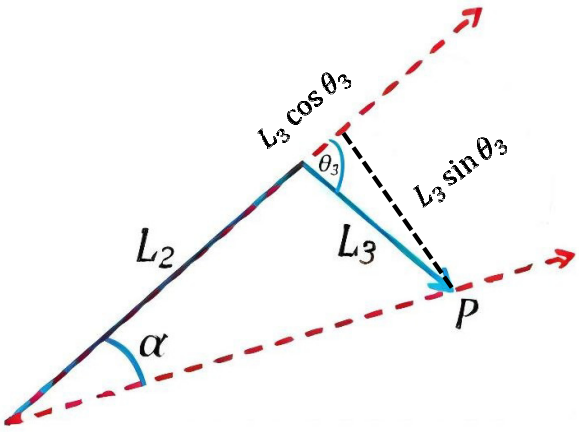
$$\boxed{\theta_3 = arc\tan\left(\frac{\sin\theta_3}{\cos\theta_3}\right)}$$

- Rumus untuk θ_2



Gambar 4.8 *Inverse Kinematic* untuk θ_2

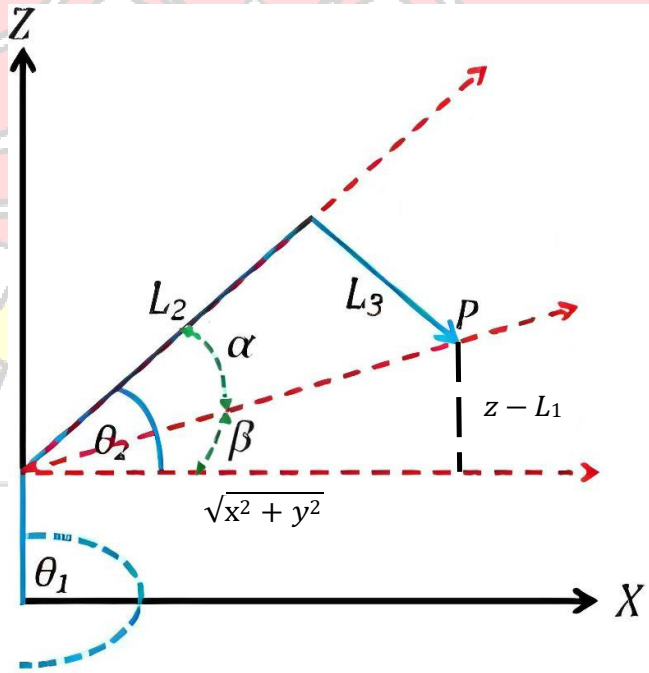
Dari gambar di atas dapat ditulis rumus sebagai berikut:



Gambar 4.9 Segitiga untuk Mencari Sudut α

$$\tan \alpha = \frac{L_3 \sin \theta_3}{L_3 \cos \theta_3 + L_2}$$

$$\alpha = arc\ tan \frac{L_3 \sin \theta_3}{L_3 \cos \theta_3 + L_2} \quad \ldots\ldots (4)$$



Gambar 4.10 Segitiga untuk Mencari Sudut β

$$\tan \beta = \frac{z - L_1}{\sqrt{x^2 + y^2}}$$

$$\beta = arc\ tan \frac{z - L_1}{\sqrt{x^2 + y^2}} \quad \ldots\ldots (5)$$

Sehingga,

$$\boxed{\theta_2 = \alpha + \beta}$$

Formulasi di atas akan digunakan untuk mendapatkan nilai θ_1, θ_2, dan θ_3 pada robot manipulator.

4.1.1.2 Setup Robot

Gambar 4.11 Mekanika Robot Manipulator Sebelum Pengembangan

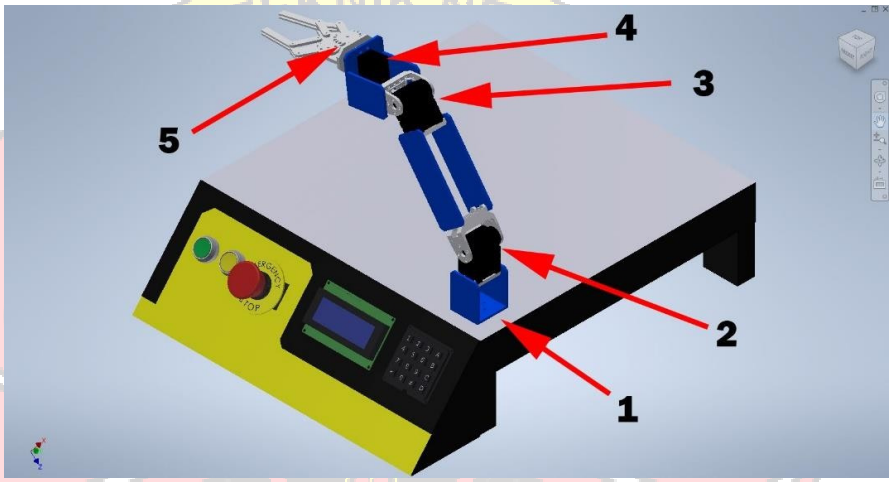
Terdapat beberapa bagian penting pada mekanika robot, yaitu:

a) Meja Manipulator

Meja manipulator memiliki ukuran panjang 400 mm, lebar 400 mm, dan

tinggi 170 mm, serta terbuat dari bahan pelat aluminium dengan ketebalan $1\frac{1}{2}$ mm. Kaki meja terbuat dari bahan aluminium hollow dengan ukuran 2,5 × 5 cm. Meja dibuat menggunakan alat-alat perkakas seperti mesin gerinda, mesin bor, tang rivet, mesin tekuk, dan lain-lain.

b) Susunan Servo



Gambar 4.12 Susunan Servo

Robot manipulator ini menggunakan 5 servo dengan konfigurasi sebagai berikut:

- **Servo 1:** Dynamixel AX-12A

  (ID: 001 | *Baud rate*: 1 Mbps | Versi protokol: 1.0)

  Limit putaran CW: 500

  Limit putaran CCW: 810

- **Servo 2:** Dynamixel AX-12A

  (ID: 002 | *Baud rate*: 1 Mbps | Versi protokol: 1.0)

  Limit putaran CW: 165

  Limit putaran CCW: 450

- **Servo 3:** Dynamixel AX-12A

  (ID: 003 | *Baud rate*: 1 Mbps | Versi Protokol: 1.0)

  Limit putaran CW: 215

  Limit CCW: 790

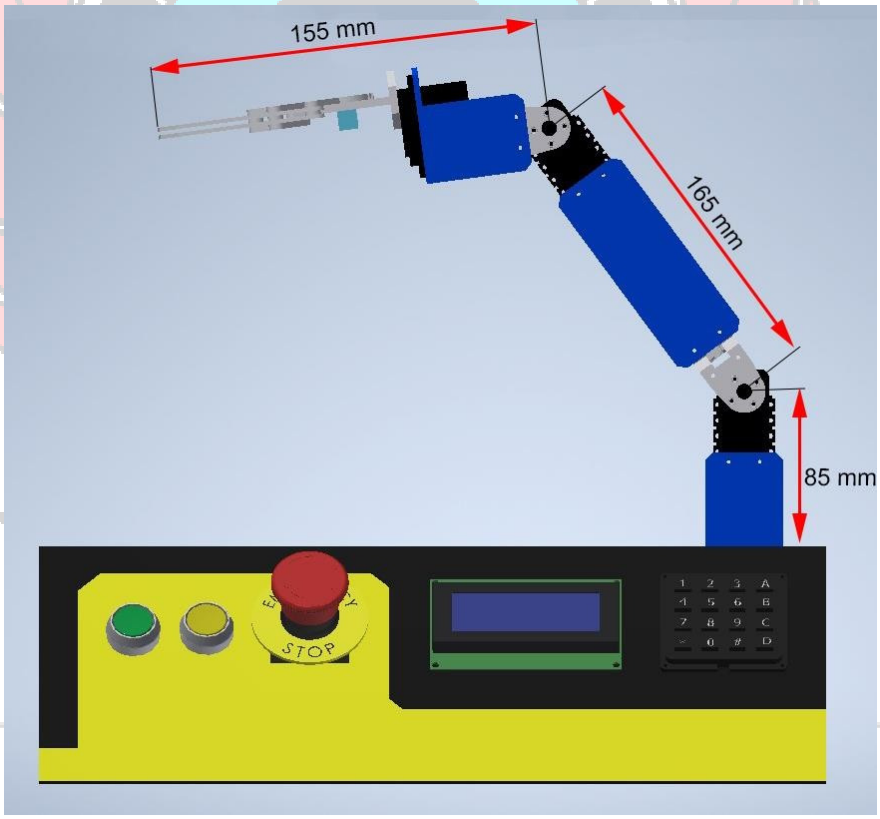
- **Servo 4:** Tower Pro MG996R

  Jangkauan servo: 0 - 180 derajat | Posisi terbaik: 90

- **Servo 5:** Tower Pro SG90

  Jangkauan terbaik servo: 120 - 180

c) Struktur *Link*



Gambar 4.13 Struktur *Link*

- Panjang *Link*-1 (L_1): 85 mm

- Panjang *Link*-2 (L_2): 165 mm
- Panjang *Link*-3 (L_3): 155 mm

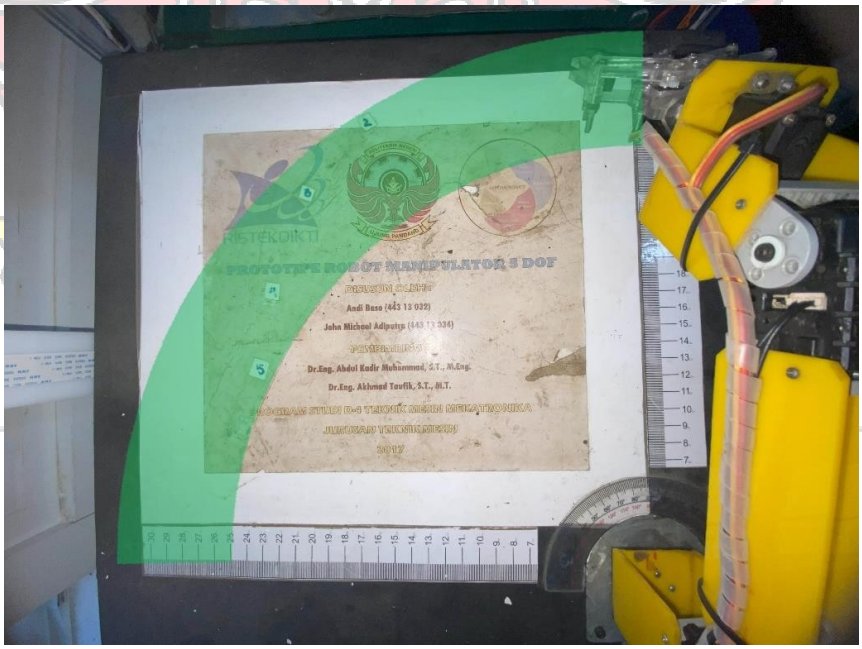
d) Setup Kamera

Kamera menggunakan teknologi terbaru, yaitu Raspberry Pi *Camera Module* 3 - 12MP dengan *Auto Focus* dan *Wide-Angle Lens*. Kamera ini terhubung ke Raspberry Pi dengan kabel sepanjang 1 meter. Tiang kamera terbuat dari bahan akrilik dengan warna susu setebal 3 mm dan 4 mm.

e) Luas Bidang Kerja (*Working Area)*

Luas bidang kerja robot merupakan luas / area yang dapat dijangkau oleh robot dan menjadi bidang kerja robot, serta menjadi acuan untuk menentukan *set point*, *target point*, maupun *end point* dari pergerakan robot.

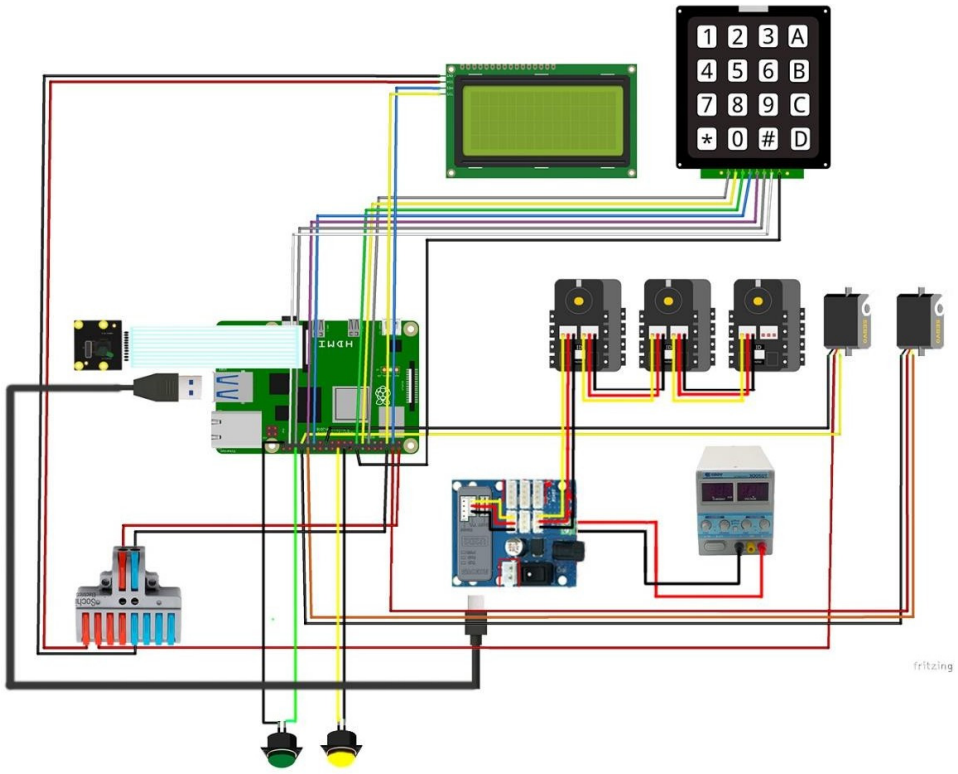
$$Luas\ Bidang\ Kerja = 33574{,}56\ mm^2 = 335{,}7456\ cm^2$$



Gambar 4.14 Luas Bidang Kerja Robot Manipulator

4.1.2 Hasil Pekerjaan Elektronika

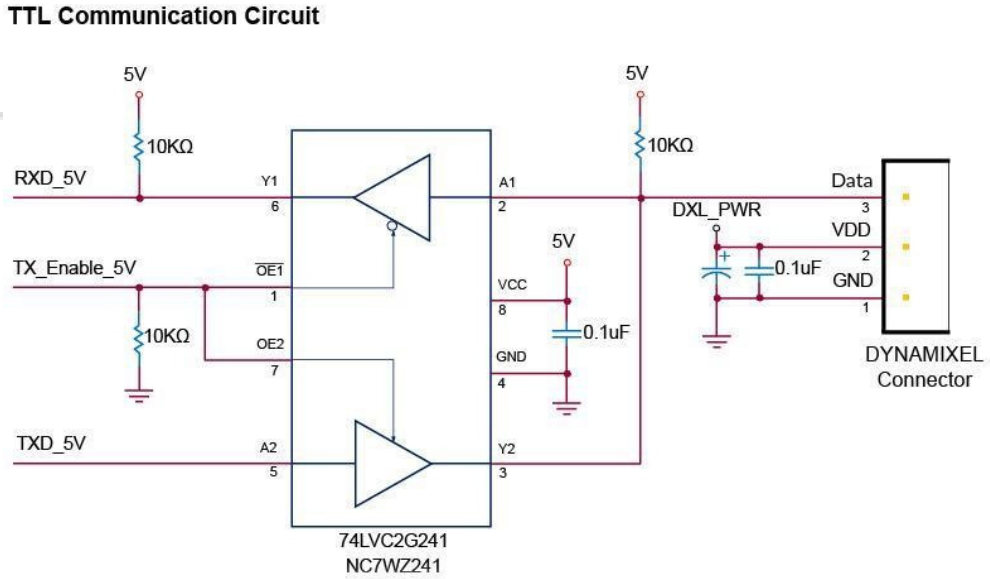
Pada sistem elektronika, proyek ini menggunakan sebuah *Single Board Computer* (SBC) yaitu Raspberry Pi 4 Model B dengan RAM 8 Giga byte. Berikut adalah gambar rangkaian elektronika robot.



Gambar 4.15 Rangkaian Elektronika Robot Manipulator

Pada rangkaian tersebut juga dapat dilihat bahwa robot ini menggunakan sebuah komponen yang disebut USB *Communication Converter* U2D2 yang dipasangkan dengan U2D2 *Power Hub Board*. Komponen ini berfungsi sebagai konverter untuk mengubah sinyal UART yang diberikan Raspberry Pi menjadi sinyal *half-duplex* yang akan diterima oleh Dynamixel AX-12A. Komunikasi ini juga sering disebut sebagai komunikasi TTL.

Berikut adalah rangkaian komunikasi TTL komponen USB *Communication Converter* U2D2:



Gambar 4.16 Rangkaian Komunikasi TTL U2D2

Robot ini juga menggunakan 2 sumber daya, yaitu:

- Ke Raspberry Pi: 5V DC melalui USB-C konektor
- Ke U2D2 *Power Hub*: 12V DC melalui soket terminal

4.1.3 Hasil Pekerjaan Informatika

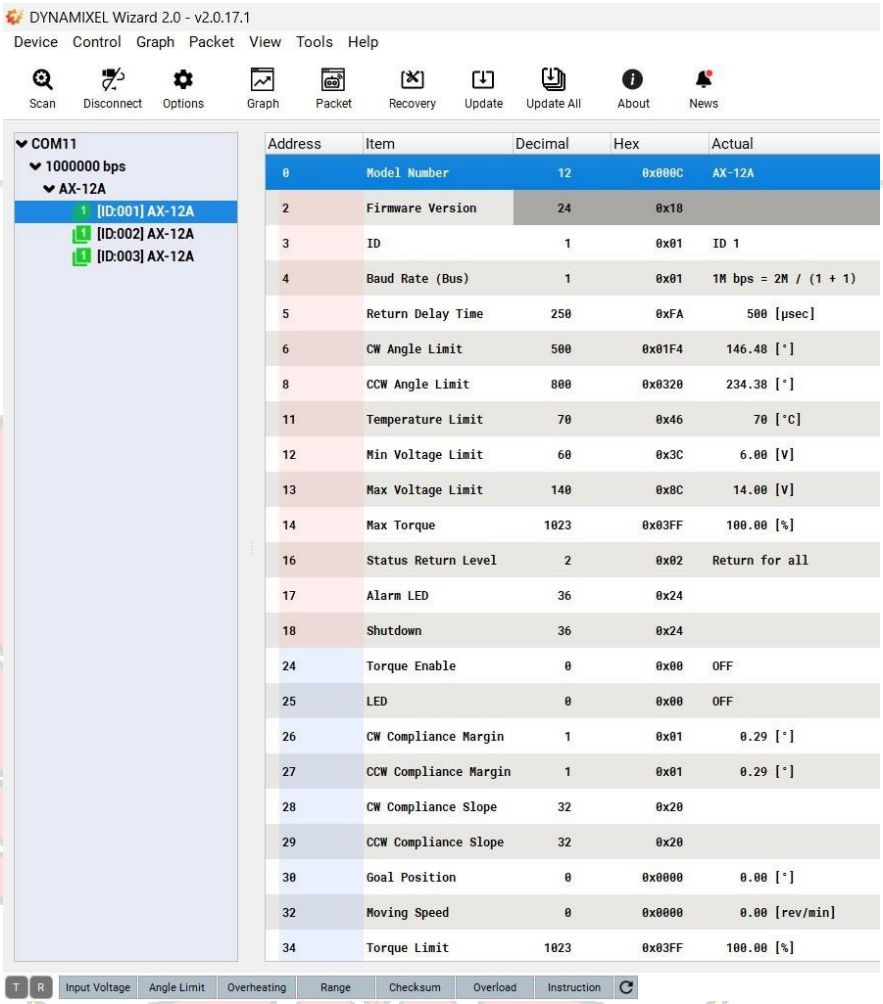
4.1.3.1 Pengaturan Dynamixel AX-12A

Install Software Dynamixel Wizard 2.0 pada windows. *Software* dapat diunduh melalui *link* berikut:

https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_wizard2/

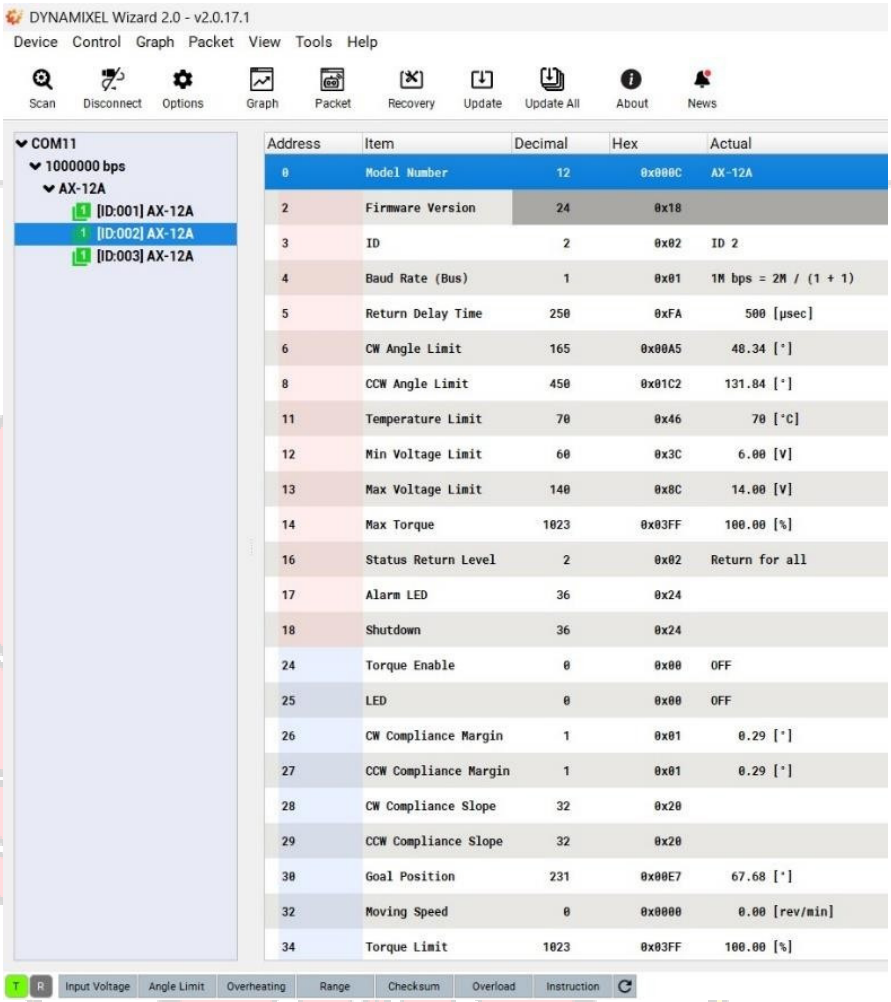
Setelah itu lakukan *update firmware* dan lakukan konfigurasi pengaturan dynamixel seperti berikut:

a) Dynamixel-001



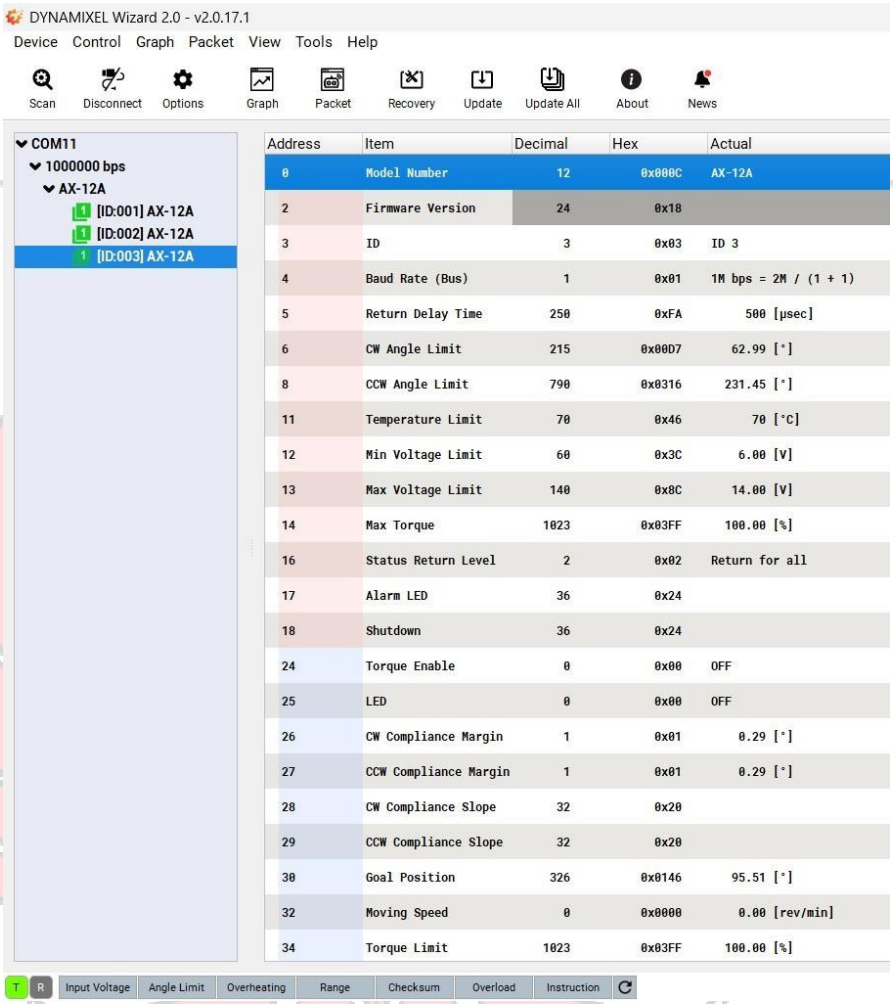
Gambar 4.17 Konfigurasi Dynamixel AX-12A ID-001

b) Dynamixel-002



Gambar 4.18 Konfigurasi Dynamixel AX-12A ID-002

c) Dynamixel-003



Gambar 4.19 Konfigurasi Dynamixel AX-12A ID-003

4.1.3.2 Deep Learning

Di tahap ini, digunakan teknologi atau model yang disebut sebagai YOLOv8 dari Ultralytics. Tujuan utama dari tahap ini adalah untuk menghasilkan sebuah model yang telah dilatih sesuai dengan kebutuhan robot manipulator. Model ini nantinya akan digunakan untuk melakukan prediksi deteksi obyek yang akan

dilakukan pada program utama nantinya. Untuk itu, perlu dilakukan beberapa langkah sebagai berikut:

a) *Data Collection*

Proses *deep learning* ini tidak menggunakan *dataset* yang telah tersedia di internet. Oleh karena itu, perlu dibuat *dataset*. Langkah ini bertujuan untuk mengambil gambar-gambar obyek pada meja robot manipulator dengan posisi yang berbeda-beda.

Untuk mengambil gambar, gunakan raspberry pi yang telah terhubung dengan kamera. Gambar yang diambil memiliki resolusi sebesar 1280 px × 720 px.

Berikut adalah program yang digunakan untuk mengambil gambar:

```
import cv2
from picamera2 import Picamera2

cam = Picamera2()
cam.preview_configuration.main.size=(1280,720) #
biggest resolution is (1920, 1080). It is not
recommended because it may not generate 30 fps.
cam.preview_configuration.main.format="RGB888"
cam.preview_configuration.align()
cam.configure("preview")
cam.start()

cv2.namedWindow("Python Webcam Screenshot App")

img_counter = 0

while True:
    frame=cam.capture_array()
    cv2.imshow("piCam",frame)
    k = cv2.waitKey(1)

    if k%256 == 27:
        print("Escape hit, closeing the app")
```

```
        break

    elif k%256 == 32:
        img_name = "img0000.png".format(img_counter)
        cv2.imwrite(img_name,frame)
        print("screenshot taken")
        img_counter += 1

cv2.destroyAllWindows()
```

Berikut ini adalah *sample* data berupa gambar yang diambil:

Gambar 4.20 Data Gambar dengan Nama File “img0001.png”

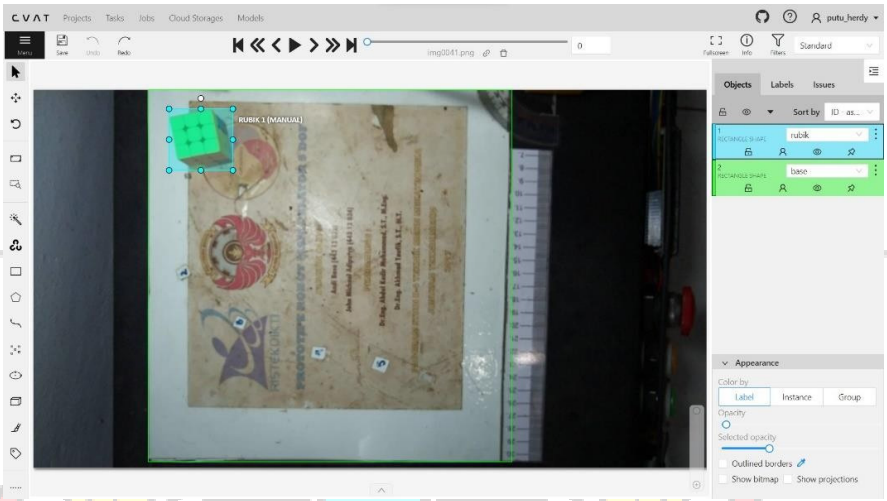
b) *Data Annotation*

Pada langkah ini, masing-masing gambar yang telah diambil kemudian dianotasi. Platform yang digunakan untuk mempermudah proses anotasi ini adalah CVAT yang dapat diakses melalui www.cvat.ai.

Terdapat 2 kelas yang dibutuhkan pada penelitian ini, yaitu:

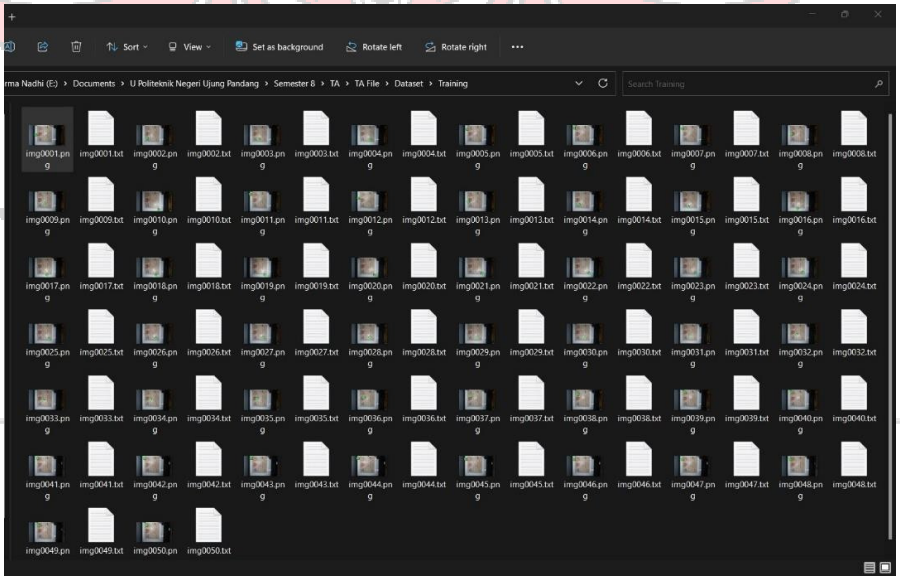
0: rubik (merupakan obyek utama yang akan dideteksi)

1: base (meja manipulator)



Gambar 4.21 Proses Anotasi Gambar

Setelah dianotasi, *export task dataset* dengan format YOLO 1.1. Kemudian masukkan *file* dengan format ".txt" ke folder yang sama dengan gambar-gambar yang telah diambil sebelumnya. Akhirnya *dataset* telah selesai dibuat. *Dataset* ini terdiri dari 50 *file* gambar dengan format ".png", serta 50 *file* anotasi dengan *format* ".txt".

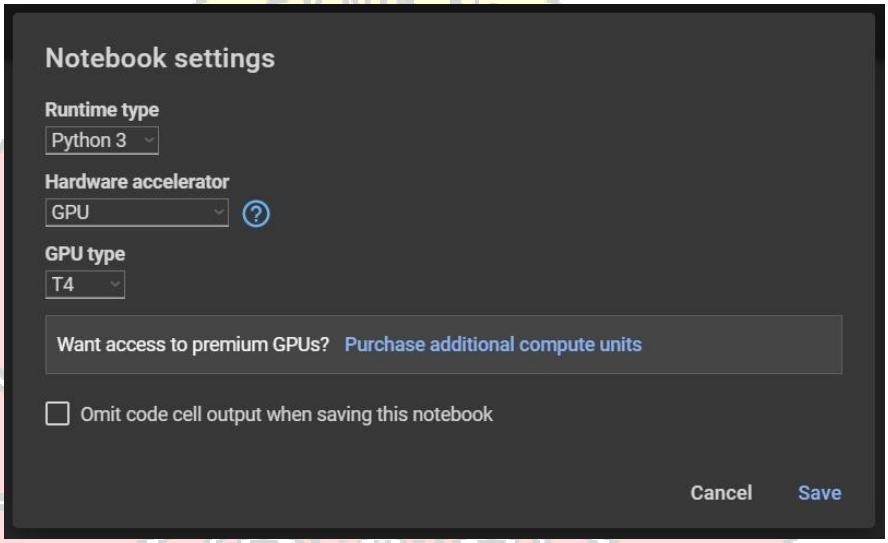


Gambar 4.22 Isi Folder *Dataset* yang Telah Dibuat

c) *Data Training*

Training data dilakukan dengan menggunakan platform Google Colaboratory. Unggah *file dataset* ke dalam satu folder di Google Drive, kemudian jalankan sel-sel berikut:

1) *Set Notebook Settings*:



Gambar 4.23 *Notebook Settings*

2) *Mount* Google Drive:

```
# Mount Goo
from google.colab import driv
drive.mount('/content/drive')
```

3) Pindah *Working Directory*:

```
# Change Working Directory
%cd '/content/drive/MyDrive/Politeknik Negeri Ujung
Pandang/Semester 8/Colab/Tutor5a_annotation/' # Edit
this relative to your path
```

4) *Install Ultralytics*:

```
# Pip install (Recommended)

!pip install ultralytics
```

5) *Import Required Libraries*:

```
## importing required libraries
import os
import shutil
import random

!pip install tqdm --upgrade
from tqdm.notebook import tqdm

import locale
locale.getpreferredencoding = lambda: "UTF-8"
```

6) *Create Folder*:

```
# Create Folder
train_path_img = "./yolo_data/images/train/"
train_path_label = "./yolo_data/labels/train/"
val_path_img = "./yolo_data/images/val/"
val_path_label = "./yolo_data/labels/val/"
test_path = "./yolo_data/test"
```

7) *Train Test Split*:

```
'''
Split the dataset into train and test and creates the
train.txt and test.tx with
the respective path of the images in each folder
'''

def train_test_split(path,neg_path=None, split =
0.2):
    print("------ PROCESS STARTED ------- ")


    files = list(set([name[:-4] for name in
os.listdir(path)])) ## removing duplicate names i.e.
counting only number of images


    print (f"--- This folder has a total number of
{len(files)} images -- ")
    random.seed(42)
    random.shuffle(files)
```

```
    test_size = int(len(files) * split)
    train_size = len(files) - test_size
    ## creating required directories

    os.makedirs(train_path_img, exist_ok = True)
    os.makedirs(train_path_label, exist_ok = True)
    os.makedirs(val_path_img, exist_ok = True)
    os.makedirs(val_path_label, exist_ok = True)


    ### ----------- copying images to train folder
    for filex in tqdm(files[:train_size]):
      if filex == 'classes':
          continue
      shutil.copy2(path + filex +
'.png',f"{train_path_img}/" + filex + '.png' )
      shutil.copy2(path + filex + '.txt',
f"{train_path_label}/" + filex + '.txt')



    print(f"------ Training data created with 80%
split {len(files[:train_size])} images ------- ")

    if neg_path:
        neg_images = list(set([name[:-4] for name in
os.listdir(neg_path)])) ## removing duplicate names
i.e. counting only number of images
        for filex in tqdm(neg_images):
            shutil.copy2(neg_path+filex+ ".png",
f"{train_path_img}/" + filex + '.png')

        print(f"------ Total {len(neg_images)}
negative images added to the training data ------- ")

        print(f"------ TOTAL Training data created
with {len(files[:train_size]) + len(neg_images)}
images ------- ")



    ### copytin images to validation folder
    for filex in tqdm(files[train_size:]):
      if filex == 'classes':
          continue
```

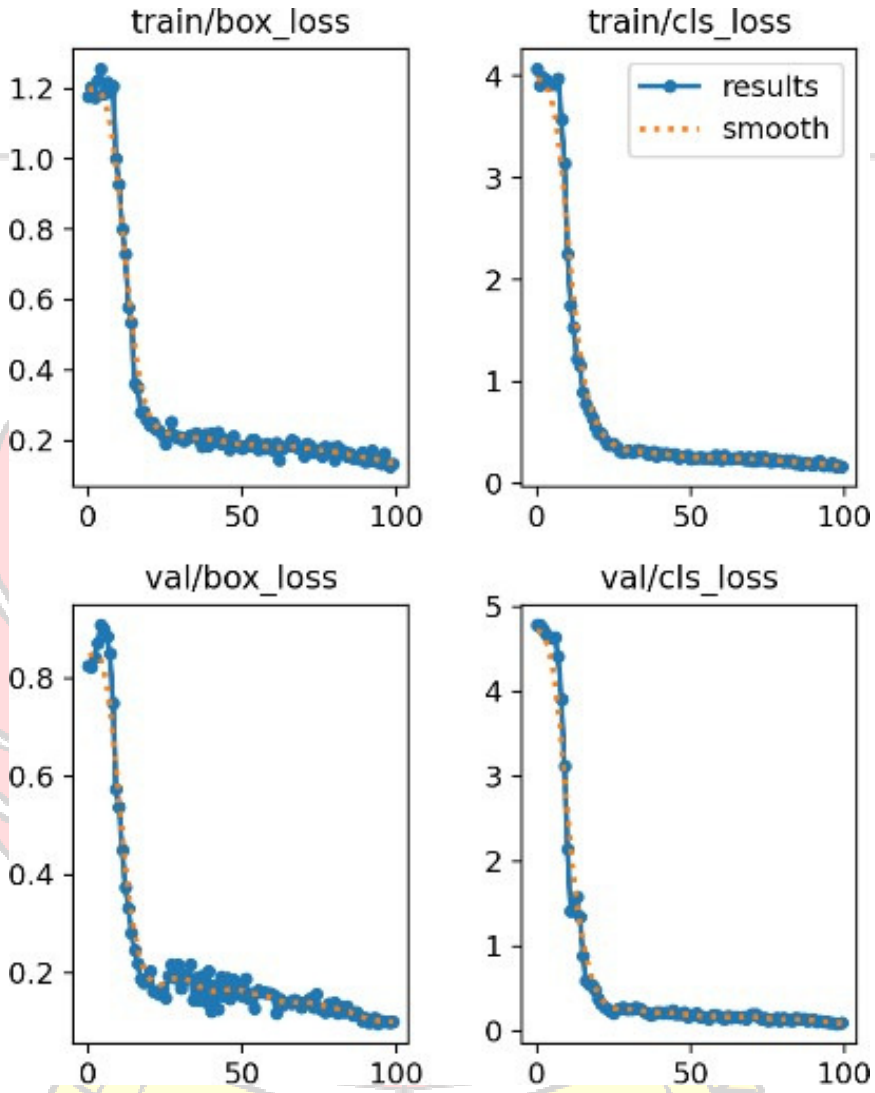
```
      # print("running")
      shutil.copy2(path + filex + '.png',
f"{val_path_img}/" + filex + '.png' )
      shutil.copy2(path + filex + '.txt',
f"{val_path_label}/" + filex + '.txt')

    print(f"------ Testing data created with a total
of {len(files[train_size:])} images ---------- ")

    print("------ TASK COMPLETED ------- ")

## spliting the data into train-test and creating
train.txt and test.txt files
#
train_test_split('/content/drive/MyDrive/custom_noteb
ooks/yolo_data/')

### for label_tag
train_test_split('/content/drive/MyDrive/Politeknik
Negeri Ujung Pandang/Semester
8/Colab/Tutor5c_training/data/') ### without negative
images
# train_test_split('./data/','./negative_images/')
### if you want to feed negative images
```

8) Training Model

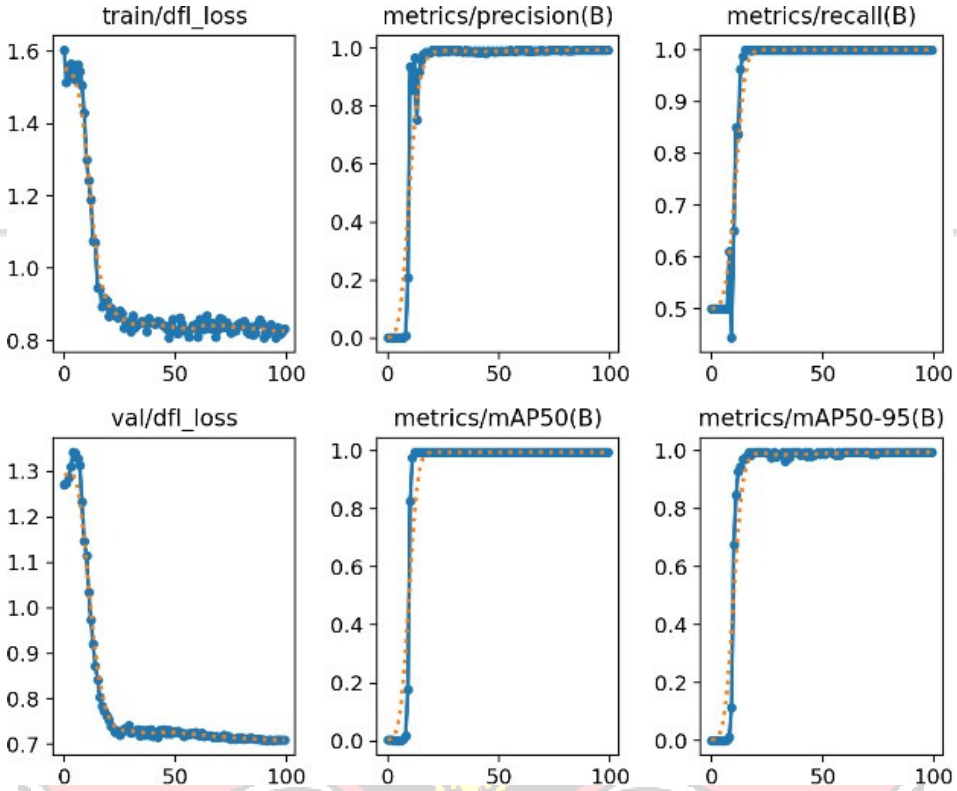
```
from ultralytics import YOLO

model = YOLO('yolov8m.pt')

# Train the model
results =
model.train(data='/content/drive/MyDrive/Politeknik
Negeri Ujung Pandang/Semester
8/Colab/Tutor5c_training/dataset.yaml', epochs=100,
imgsz=640, batch=64,
project='/content/drive/MyDrive/Politeknik Negeri
Ujung Pandang/Semester
8/Colab/Tutor5c_training/training_results',
name='rubik_detector')
```

Dengan ini, pada *folder weights* didapatkan sebuah model yang sudah di-*training* dengan nama "best.pt". Model ini nantinya akan digunakan untuk mendeteksi sebuah rubik dan meja manipulator.

Berikut ini adalah parameter hasil *training* yang diperoleh:



Gambar 4.24 Grafik Hasil *Training* (I)



Gambar 4.25 Grafik Hasil *Training* (II)

Rumus untuk perhitungan akurasi, presisi, dan recall:

$$Accuracy = \frac{True_{positive} + True_{negative}}{True_{positive} + True_{negative} + False_{positive} + False_{negative}}$$

$$Precision = \frac{True_{positive}}{True_{positive} + False_{positive}}$$

$$Recall = \frac{True_{positive}}{True_{positive} + False_{negative}}$$

Dengan menggunakan rumus di atas maka didapatkan nilai Akurasi, Presisi, dan Recall sebagai berikut:

Akurasi (mAP50)	: 0.995
Presisi	: 0.99305
Recall	: 1

4.1.3.3 Machine Learning

Pada saat pengambilan gambar (*data collection*) juga dilakukan pengambilan data berupa koordinat rubik untuk nilai x, y, dan z relatif terhadap meja manipulator dalam situasi real dengan satuan milimeter [mm]. Kemudian untuk setiap *file* data gambar dan anotasinya dicari koordinat rubik relatif terhadap basenya dalam pixel [px]. Algoritmanya adalah seperti ini:

1) *Upload file* gambar dan anotasi ke dalam *notebook runtime* lalu jalankan *cell* berikut ini:

```
import cv2 as cv
from google.colab.patches import cv2_imshow

def get_color(label):
    if(label == 0):
      return (207, 151, 0)
    elif(label == 1):
      return (0, 255, 0)

def draw_bounding_box(img):
 # label_norm, x_norm, y_norm, width_norm, height_norm
= map(float, annotation.strip().split())
  #global x_norm, y_norm, width_norm, height_norm
  global x_min_0, y_min_0, x_max_0, y_max_0
  global x_min_1, y_min_1, x_max_1, y_max_1

  label = int(label_norm)

  # Convert normalized coordinates to pixel values
  x = float(x_norm * image_width)
  y = float(y_norm * image_height)
```

```
  w = float(width_norm * image_width)
  h = float(height_norm * image_height)

  # Calculate the top-left and bottom-right coordinates
of the bounding box
  x_min = round(x - w / 2)
  y_min = round(y - h / 2)
  x_max = round(x + w / 2)
  y_max = round(y + h / 2)

#"{:.4f}".format(number)
#f"{number:.4f}" x - w / 2

  def text_pos(label):
    if(label == 0):
      return (x_min, y_min - 5)
    elif(label == 1):
      return (x_min, y_min + 17)

  if(label == 0):
    x_min_0 = x_min
    y_min_0 = y_min
    x_max_0 = x_max
    y_max_0 = y_max
  elif(label == 1):
    x_min_1 = x_min
    y_min_1 = y_min
    x_max_1 = x_max
    y_max_1 = y_max

  print(label)
  print(x, y, w, h)
  print('-')
  print(x_min, y_min, x_max, y_max)
  print(' ')

  start_point, end_point = ((x_min, y_min), (x_max,
y_max))
  cv.rectangle(img, start_point, end_point,
get_color(label), 2)

  # Put the class label text near the bounding box
  cv.putText(img, str(label), text_pos(label),
cv.FONT_HERSHEY_SIMPLEX, 0.75, get_color(label), 2)
```

```
NAME = "img0050" # change to img0002 and img0003

img = cv.imread(f"{NAME}.png")

image_height, image_width, _ = img.shape

with open(f"{NAME}.txt") as f:
    lines = f.readlines()
    for annotation in lines:
      label_norm, x_norm, y_norm, width_norm,
height_norm = map(float, annotation.strip().split())
      draw_bounding_box(img)

cv2_imshow(img)
#cv.imwrite(f"{NAME}_output.png", img)

cv.waitKey(0)
cv.destroyAllWindows()

x_img = (x_min_0 + (x_max_0 - x_min_0) / 2) - x_min_1
y_img = y_max_1 - (y_min_0 + (y_max_0 - y_min_0) / 2)

print(x_img, y_img)
```

Maka didapatkan nilai x_img dan y_img yang merupakan koordinat rubik relatif terhadap base manipulator dalam gambar dengan satuan pixel. Kemudian buat sebuah dataset yang berisi kolom seperti berikut:

1) Nama file gambar: berisi “img0001.png” sampai dengan “img0050.png”.
2) x_img: koordinat relatif terhadap sumbu x dalam gambar.
3) y_img: koordinat relatif terhadap sumbu y dalam gambar.
4) x_manip: koordinat relatif terhadap sumbu x meja manipulator yang sebenarnya.
5) y_manip: koordinat relatif terhadap sumbu y meja manipulator yang sebenarnya.

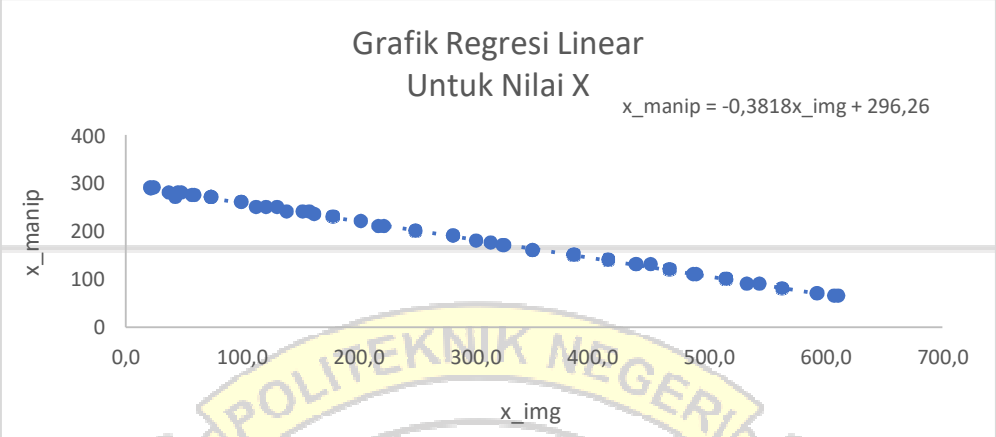
Pada proyek ini digunakan SQL XAMPP untuk membuat *dataset*. Berikut adalah gambar tabel yang telah dibuat:

image	x_img	y_img	x_manip	y_manip
img0001	43.500000	363.500000	270	155
img0002	161.500000	251.500000	235	200
img0003	313.000000	197.000000	175	220
img0004	450.000000	168.500000	130	230
img0005	57.000000	556.500000	275	92
img0006	59.000000	603.500000	275	70
img0007	73.500000	583.000000	270	80
img0008	138.000000	400.500000	240	150
img0009	323.000000	285.500000	170	190
img0010	592.000000	47.500000	70	280
img0011	21.500000	600.500000	288	70
img0012	37.000000	462.000000	280	125
img0013	112.000000	329.500000	250	175
img0014	216.500000	237.500000	210	210
img0015	280.500000	153.500000	190	240

Gambar 4.26 *Dataset* Berisi Tabel yang Dibuat dengan SQL

Setelah itu *export dataset* yang telah dibuat ke dalam format ".csv".

Jika data tersebut dianalisis dengan menggunakan Scikit-Learn atau Microsoft Excel, maka akan didapat grafik dan *summary* sebagai berikut:



Gambar 4.27 Grafik Regresi Linear untuk Koordinat x

SUMMARY OUTPUT FOR X VALUES

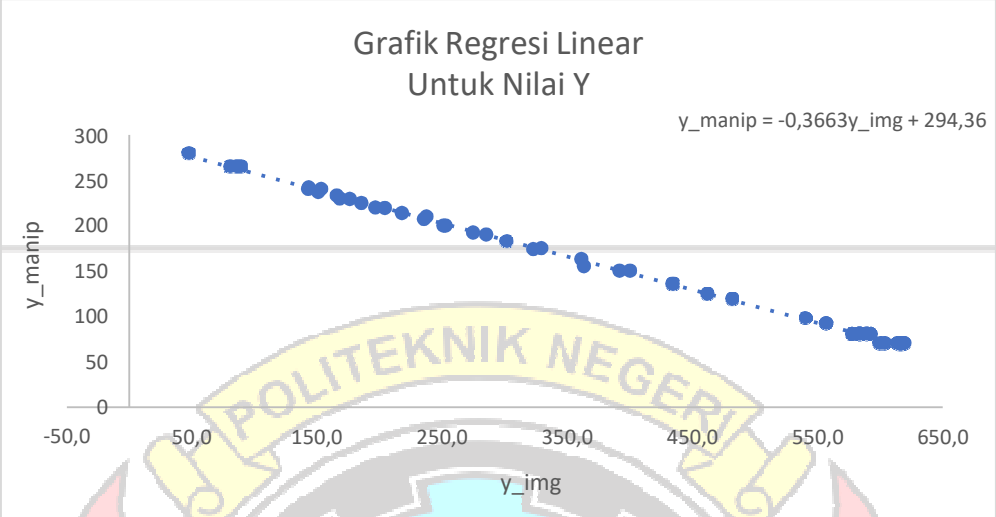
Regression Statistics	
Multiple R	0,99941654
R Square	0,998833421
Adjusted R Square	0,998809117
Standard Error	2,526391356
Observations	50

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	1	262313,9526	262313,9526	41097,94798	4,62638E-72
Residual	48	306,3673577	6,382653284		
Total	49	262620,32			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Low*
Intercept	296,2609004	0,619915672	477,9051635	6			
X_img	-0,						

Gambar 4.28 *Summary Output* untuk Nilai Koordinat x



Gambar 4.29 Grafik Regresi Linear untuk Koordinat y

SUMMARY OUTPUT FOR Y VALUES

Regression Statistics	
Multiple R	0,999601009
R Square	0,999202178
Adjusted R Square	0,999185557
Standard Error	2,064172127
Observations	50

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	1	256141,8013	256141,8013	60115,80132	5,06953E-76
Residual	48	204,5187154	4,260806571		
Total	49	256346,32			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95,0%*	*Upper 95,0%*
Intercept	294,3671088	0,608575593	483,6985117	3,52504E-90	293,1434856	295,590732	293,1434856	295,590732
X Variable 1	-0,366321896	0,001494062	-245,1852388	5,06953E-76	-0,369325908	-0,363317883	-0,369325908	-0,363317883

Gambar 4.30 *Summary Output* untuk Nilai Koordinat y

4.1.3.4 Raspberry Pi

Sebelum menjalankan program terdapat beberapa *library* yang harus disiapkan / di-*install*, yaitu:

4.1 Paket atau *Library* yang Dibutuhkan

Nama Paket / Library	**Cara Install**
Python 3	$ sudo apt-get install python3
pip	$ sudo apt install python3-pip
Dynamixel SDK	$ git clone https://github.com/ROBOTIS-GIT/DynamixelSDK.git
Ultralytics	$ pip install ultralytics

Selain itu, berikut ini adalah potongan program-program python yang dibutuhkan untuk menjalankan robot. Lebih lengkap dapat dilihat pada bagian lampiran.

1) manip_ai.py

```
##################################
##### DEEP LEARNING PROCESS #####
##################################

model = YOLO('best_002.pt')  # Load model

# Predict captured image using the model
results = model(source='img0000.png',
                conf=0.55,
                save=True,
                save_txt=True,
                show=True,
```

```
                show_labels=True)  # predict on an image

for result in results:
    boxes = result.boxes  # Boxes object for bbox outputs
    masks = result.masks  # Masks object for segmentation
masks outputs
    probs = result.probs  # Class probabilities for
classification outputs

tensor = boxes.xyxy.clone().detach().requires_grad_(True)
x_img = (tensor[1,0].item() + (tensor[1,2].item() -
tensor[1,0].item()) / 2) - tensor[0,0].item()
y_img = tensor[0,3].item() - (tensor[1,1].item() +
(tensor[1,3].item() - tensor[1,1].item()) / 2)

print("x_img: %.1f" % (x_img))
print("y_img: %.1f" % (y_img))
```

2) manip_manual.py

```
def Formulasi():
    global L1, L2, L3
    global h, Z
    global X, Y, theta1
    global c3, s3, s3a, theta3
    global theta2
    global T1a, T2a, T3a
    global T1, T2, T3

    L1 = 85     # mm
    L2 = 165    # mm
    L3 = 155    # mm

    # Formulasi Invers Kinematics Robot Manipulator
    theta1 = math.atan2(Y, X)    # Radian

    h = Z - L1  # mm    # This is for theta3
    c3 = (X*X + Y*Y + h*h - L2*L2 - L3*L3) / (2*L2*L3)  #
This is cos theta 3
    s3 = -math.sqrt(1 - c3*c3) # For down elbow
    s3a = math.sqrt(1 - c3*c3) # For up elbow
    theta3 = math.atan2(s3, c3) # Radian

    alfa = math.atan2(h , math.sqrt(X*X + Y*Y))
```

```
    beta = math.atan2(math.sin(math.acos(c3)) * L3, L2 +
math.cos(math.acos(c3)) * 155)
    theta2 = alfa + beta

    T1a = theta1 * 180.0 / math.pi  # Degree    # T1a is
theta1 in Degree
    T2a = theta2 * 180.0 / math.pi  # Degree    # T2a is
theta2 in Degree
    T3a = theta3 * 180.0 / math.pi  # Degree    # T3a is
theta3 in Degree

    T1 = 510 + (T1a / 0.29297)
    T2 = 204 + (T2a / 0.29297)
    T3 = 460 + (T3a / 0.29297)
```

4.1.5 Pengujian

4.1.5.1 Pengujian Servo

Berikut adalah rumus yang digunakan untuk menghitung error pergeseran nilai bit sinyal posisi servo. Nilai sinyal ini menandakan posisi servo saat ini (*present position*) yang disimbolkan dengan $\mathbf{T1}_{Act}$, $\mathbf{T2}_{Act}$, dan $\mathbf{T3}_{Act}$ yang berarti nilai posisi aktual servo dynamixel AX-12A. *Range* posisi servo *default* adalah 0-1023.

$$Mean\ Absolut\ Error\ (MAE) = \frac{1}{n}\sum_{i=1}^{n}|Ti_{Act} - Ti|$$

$$Mean\ Absolut\ Error\ (\%) = \left(\frac{1}{n}\sum_{i=1}^{n}|Ti_{Act} - Ti|\right) \div 1023 \times 100$$

Tabel 4.2 Pengujian Servo Dynamixel AX-12A

Eksperimen	Kalkulasi Matematis			Aktual			Mean Absolut Error (%)
	T1	T2	T3	$T1_{Act}$	$T2_{Act}$	$T3_{Act}$	
1	617	225	406	613	217	402	0,52
2	649	222	411	646	213	406	0,55
3	692	278	288	689	266	284	0,62
4	723	299	237	719	287	234	0,62
5	572	273	300	569	262	298	0,52
6	771	276	292	766	264	289	0,65
7	770	287	266	767	278	262	0,52
8	691	303	225	687	292	225	0,49
9	561	301	229	558	293	227	0,42
10	566	276	282	563	266	281	0,46
11	557	279	286	555	275	291	0,36
12	615	270	300	611	264	298	0,36
13	670	304	218	665	296	216	0,49
14	558	265	312	557	259	309	0,33
15	587	246	355	585	239	351	0,42
16	628	243	362	624	236	358	0,49
17	660	262	319	655	255	316	0,49
18	683	245	357	679	237	353	0,52
19	710	242	363	707	234	359	0,49
20	727	266	310	723	258	306	0,52
Mean Total Absolut Error							**0,49 %**

Data detail mengenai pengujian servo dynamixel dapat dilihat di github peneliti: Vkurpmax

4.1.5.2 Energi yang Digunakan

Tabel 4.3 Energi yang Digunakan Robot

Eksperimen	**Menit Pencatatan**	**Tegangan [V]**	**Arus [A]**	***Power Factor***	**Daya Tertinggi [Watt]**	**Total Energi [kWh]**
0 (*Idle*)	2	232	0,145	0,61	22,7	0,000
1	4	~ 232	0,230	0,67	37,6	0,001
2	7	~ 232	~ 0,230	~ 0,67	36	0,002
3	9	~ 232	~ 0,230	~ 0,67	~ 36	0,003
4	10	~ 232	~ 0,230	~ 0,67	30	0,004

4.2 Pembahasan

4.2.1 Mekanika

- Terdapat penambahan sebuah tiang dudukan atau *mounting bracket* kamera raspberry pi.
- Mengganti alas *link*-1 yang sebelumnya menggunakan kertas menjadi sebuah pelat tipis yang dilapisi pelumas untuk mengurangi gaya gesek antara *link*-1 dan *base* meja manipulator.
- Membongkar servo-3 (Dynamixel AX-12A), membersihkan kotoran servo, dan menambahkan grease pada roda giginya sebagai bagian dari perawatan untuk memperlambat keausan.

4.2.2 Elektronika

- Mengganti servo digital TowerPro SG-90 dengan servo analog EMAX

ES08MA II. Servo towerPro SG-90 tidak cukup untuk mengakomodasi sistem *gripper* akibat dari sistem sinyal keluaran yang dihasilkan oleh Raspberry Pi berbeda dengan Arduino Mega yang digunakan pada penelitian sebelumnya.

- Mengganti kabel-kabel yang usang dan tidak berfungsi dengan baik dengan kabel baru.
- Menambahkan beberapa komponen penting seperti U2D2 USB *Communication Converter*, PHB Set, dan PCT terminal guna mendukung sistem dengan *Single Board Computer* Raspberry Pi.
- Menambahkan konektor *pin header female* pada koneksi LCD 2004 dengan I2C untuk meningkatkan fleksibilitas.

4.2.3 Sistem Kontrol

- Algoritma proses kontrol robot manipulator berhasil ditemukan oleh peneliti. Algoritma yang telah dibuat terbukti efektif diimplementasikan ke dalam sistem robot manipulator.
- Menggunakan *custom dataset* sangat efisien. *Dataset* ini dibuat sendiri dan disesuaikan dengan kebutuhan penggunaan robot. *Dataset* yang dibuat sesuai kebutuhan ini sangat berkontribusi pada peningkatan kualitas model yang dilatih. Terbukti dengan hanya menggunakan 50 data gambar dan anotasinya dapat meningkatkan akurasi (mAP50) hingga 99,5 %.
- Menggunakan model dan teknologi yang relevan sangat membantu kemudahan setiap proses *deep learning*. *You Only Look Once* (YOLO) dikembangkan terus menerus oleh peneliti serta pengembang di berbagai

dunia. YOLOv8 diluncurkan pada bulan Januari 2023. Proyek ini menggunakan *pre-trained model* yolov8m.pt dengan 100 epochs, imgsize 640, serta batch size 64. Konfigurasi ini adalah konfigurasi terbaik yang didapatkan pada penelitian ini.

- Pada proses *machine learning*, penelitian ini menggunakan 1 *database* dan 1 tabel yang berisi 2 jenis data yang berbeda, yaitu untuk koordinat x dan koordinat y. Masing-masing jenis data memiliki 2 *feature* dan 1 label. Terdapat 50 data di dalam *database*. Penelitian ini termasuk dalam kategori *supervised learning*, serta menggunakan algoritma regresi linear dengan bantuan *library* scikit-learn.
- Menggunakan *software* Real VNC Viewer, putty, zenmap dan cmd untuk menjalankan protokol *secure shell* (SSH). SSH adalah protokol untuk melakukan transfer dan mengontrol perangkat dari jarak jauh. Penelitian ini menggunakan SSH untuk mengontrol serta memonitoring keseluruhan proses robot manipulator.

BAB V KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil dan pembahasan mengenai penelitian Pengembangan Prototipe Robot Manipulator dan Sistem Monitoring berbasis *Deep Learning*, dapat disimpulkan:

1. Formulasi robot manipulator 3 DOF telah berhasil dan sesuai dengan kaidah *inverse kinematic* pada ilmu kinematika dan dinamika robot, sehingga dapat diimplementasikan ke dalam sistem prototipe robot manipulator 3 DOF.
2. Sistem mekanika, elektronika, dan sistem kontrol robot masing-masing telah dilakukan pengembangan untuk meningkatkan kinerja robot. Terbukti dengan nilai hasil uji total *error* servo dynamixel AX-12A yang lebih rendah dari penelitian sebelumnya, yang sebelumnya 0,77% menjadi 0,49%.
3. Hasil pengujian menunjukkan bahwa prototipe robot manipulator 3 DOF yang telah dikembangkan dengan mengadopsi teknologi *deep learning* mampu meningkatkan kinerja sistem robot manipulator. Terbukti dengan nilai *error* servo dynamixel AX-12A sebesar 0,49%.
4. Dengan sumber daya yang digunakan berupa arus bolak-balik (AC), daya yang digunakan rata-rata ~ 35 watt, serta total energi sebesar ~ 0,001 kWh untuk setiap 1 kali eksperimen.

5.2 Saran

Setelah melakukan penelitian Pengembangan Prototipe Robot Manipulator 3 DOF ini, terdapat beberapa saran yang ingin penulis sampaikan, yaitu:

1. Robot manipulator ini memiliki *working volume* yang sempit, sehingga perlu untuk memperhatikan penempatan objek yang akan dipindahkan.
2. Servo Dynamixel AX-12A yang digunakan sebelumnya telah mengalami penurusan torsi, peningkatan keausan, *life time* yang semakin berkurang. Dari ketiga servo, telah dilakukan penggantian pada servo nomor 2. Untuk meningkatkan kemampuan servo pada sistem disarankan untuk melakukan penggantian servo nomor 1 dan 3.
3. Sistem robot ini dapat dikembangkan dengan melakukan penambahan *dataset* untuk meningkatkan akurasi *deep learning*, serta perbaikan pada *dataset machine learning* untuk regresi linear.
4. Sistem robot ini dapat dikembangkan dengan menambahkan *Internet of Things* (IoT) dengan kualitas yang setara dengan *Industrial Internet of Things* (IIoT) dengan protokol-protokol yang lebih canggih di masa depan.
5. Menambahkan *Graphical user interface* (GUI) pada tampilan *remote device* seperti komputer atau laptop. GUI berfungsi untuk membuat tampilan sistem robot pada *remote device* menjadi lebih rapi, tertata, mudah dilihat, dan mudah dipahami.

DAFTAR PUSTAKA

Baso, Andi dan John Michael Adiputra. 2017. Pengembangan Prototipe Robot Manipulator. Skripsi. Makassar: Politeknik Negeri Ujung Pandang.

Bernier, Catherine. 2021. Robot Grippers and End Effectors: Uses, Benefits, and Cost Analysis. (*Online*). (https://howtorobot.com/expert-insight/robot-end-effectors), Diakses 8 Februari 2023.

Cahyono dkk. 2015. Kontrol Lengan Robot Badminton dan Analisis Kinematika. Dalam *Elkolind*, (2): 73.

Caysar, Dina. 2014. Pengaturan Pergerakan Robot Lengan Smart Arm Robotic AX-12A melalui Pendekatan Geometry Based Kinematic Menggunakan Arduino. Dalam *Jurnal Dina Caysar*, (2): 3.

Components 101. 2019. MG996R Servo Motor. (*Online*). (https://components101.com/motors/mg996r-servo-motor-datasheet), Diakses 11 Februari 2023.

EMAX. 2020. EMAX ES08MA II 12g Mini Metal Gear Analog Servo for RC Model & Robot PWM servo. (*Online*). (https://emaxmodel.com/products/emax-es08ma-ii-12g-mini-metal-gear-analog-servo-for-rc-model-robot-pwm-servo), Diakses 20 Juli 2023.

Genesis System. 2023. How the Right Robotic End-Effector Impacts Robotic Performance. (*Online*). (https://www.genesis-systems.com/blog/benefits-finding-right-robotic-end-effector), Diakses 8 Februari 2023.

Goryanina, Ksenia I. et al. 2018. *Review of Robotic Manipulators and Identification of the Main Problems*. Rostov-on-Don: Don State Technical University.

Herizon dan Ade Diana. 2014. Implementasi Persamaan Kinematik Maju pada Robot Manipulator. Dalam *Elektron*, (6): 67.

International Federation of Robotics. 2022. Industrial Robots. Majalah *World Robotics 2022*, (10): 9. Frankfurt.

King-sun Fu, 1987. *Robotics: Control, Sensing, Vision, and Intelligence*. United States of America: McGraw-Hill.

Mason, Matthew T. 2001. *Mechanics of Robotic Manipulation*. London: The MIT Press.

Muhammad, Abdul K. dkk. 2016. Rancang Bangun Prototipe Robot Manipulator. Makassar: Politeknik Negeri Ujung Pandang.

Mustarin, Jamil dan Miansari Mogot. 2016. Rancang Bangun Prototipe Robot Manipulator. Skripsi. Makassar: Politeknik Negeri Ujung Pandang.

Muslimin, Selamat dkk. 2015. Penerapan Flex-Sensor pada Lengan Robot Berjari Pengikut Gerak Lengan Manusia Berbasis Mikrokontroler. Dalam *Technologic*, (5): 8.

Rahmat, Ajang. 2019. Mengatasi Error pada NodeMCU Amica. (*Online*). (https://kelasrobot.com/mengatasi-error-pada-nodemcu-amica/), Diakses 7 Februari 2023.

Raspberry Pi. 2019. Raspberry Pi 4 Model B. (*Online*). (https://www.raspberrypi.com/products/raspberry-pi-4-model-b/), Diakses 6 Februari 2023.

Robotis. 2022. AX-12A. (*Online*). (https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/), Diakses 7 Februari 2023.

---------- . 2023. Dynamixel_Shield. (*Online*). (https://emanual.robotis.com/docs/en/parts/interface/dynamixel_shield/), Diakses 7 Februari 2023.

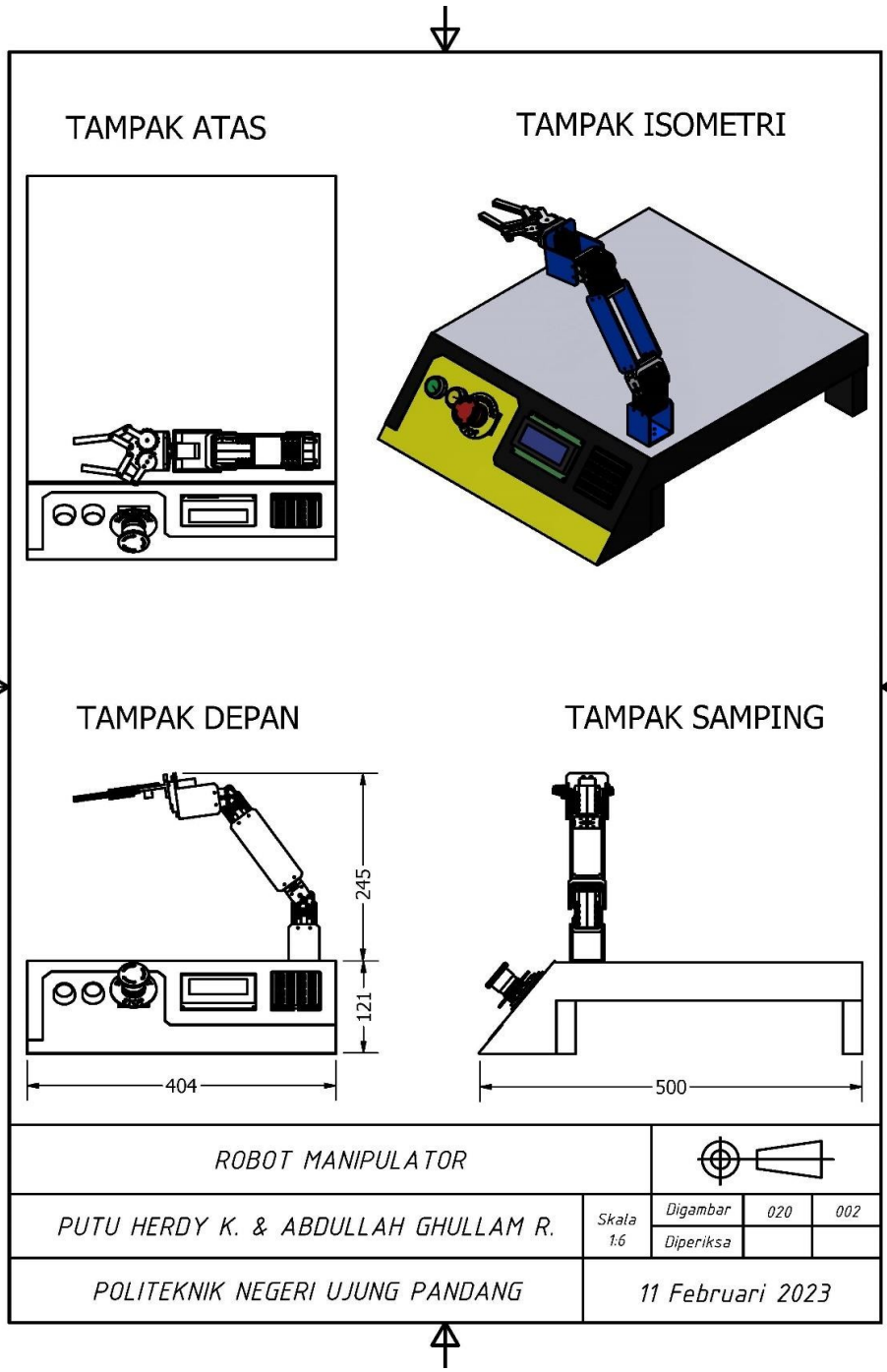
Roboticslearning. 2020. Inverse Kinematics. (*Online*). (https://www.rosroboticslearning.com/inverse-kinematics), Diakses 15 Mei 2023.

Taufik dkk. 2017. Rancang Bangun Prototipe Robot Manipulator untuk Media Praktikum. Prosiding *Seminar Hasil Penelitian (SNP2M)*. 205. Makassar: Politeknik Negeri Ujung Pandang.

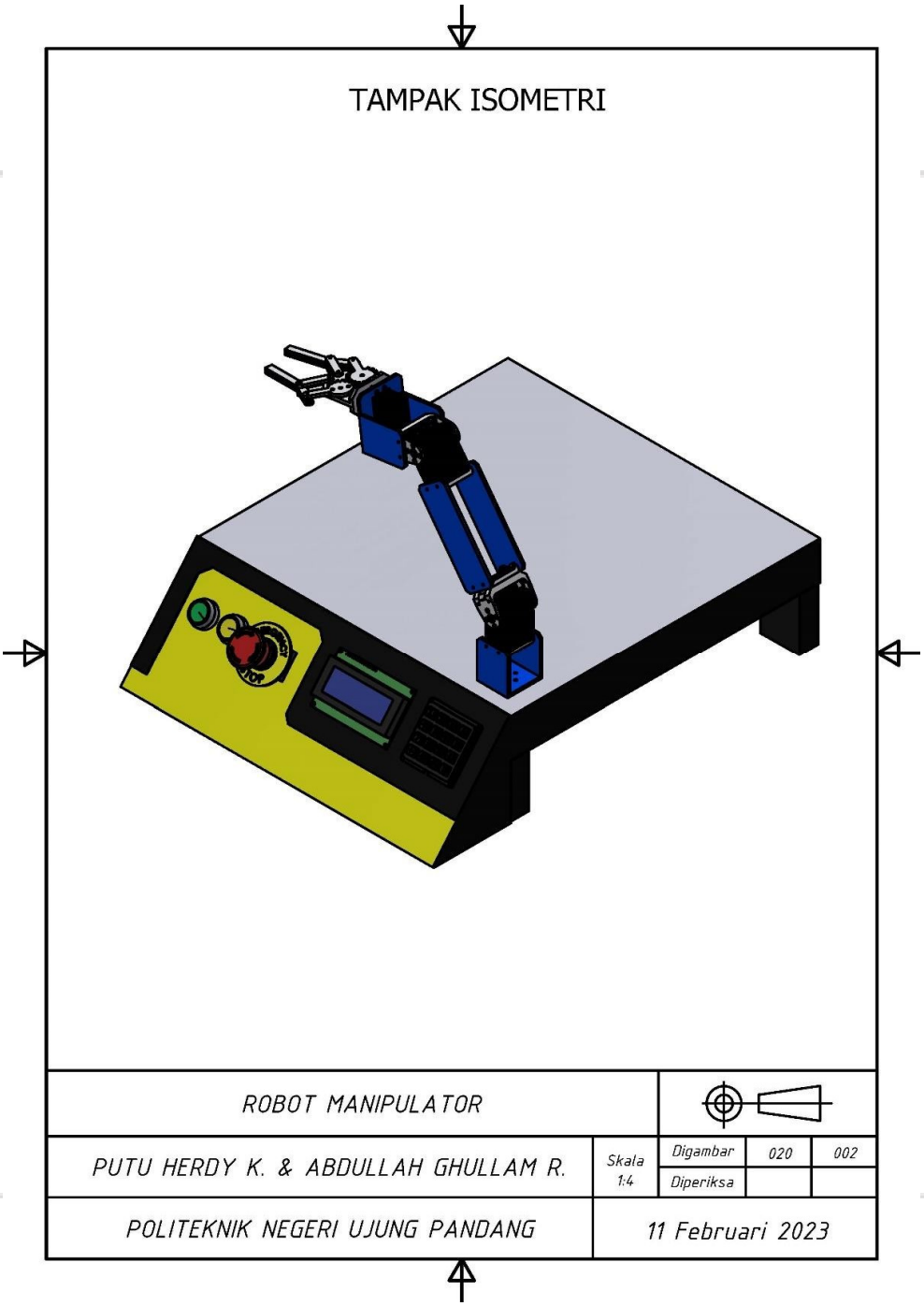
Tedrake, Russ. 2022. Robotic Manipulation: Perception, Planning, and Control. *Course MIT 6.4210/6.4212 – Robotic Manipulation*. (*Online*). (https://manipulation.mit.edu/intro.html), Diakses 29 Januari 2023.

LAMPIRAN

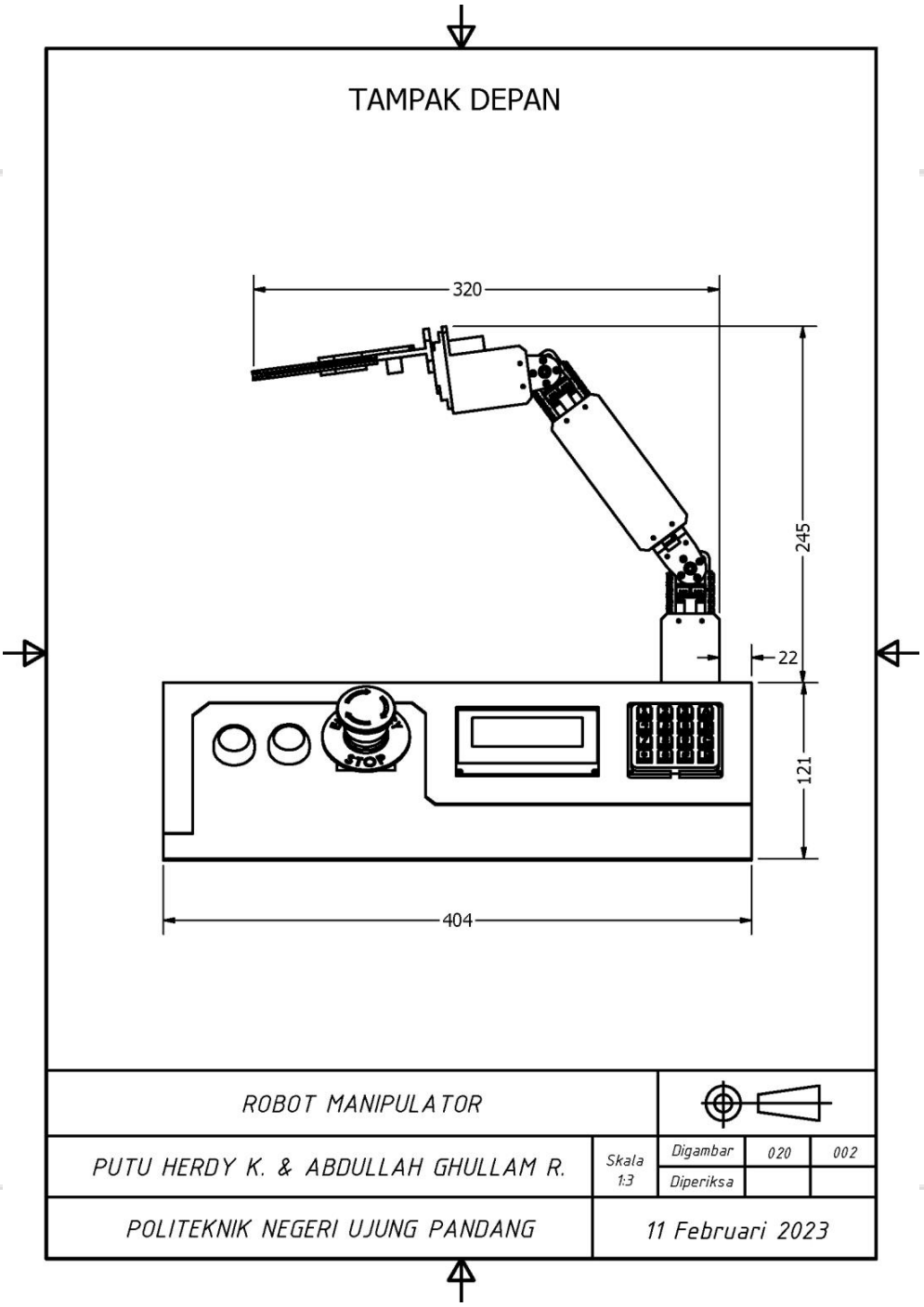
Lampiran 1 Gambar Teknik Desain Awal Robot Manipulator



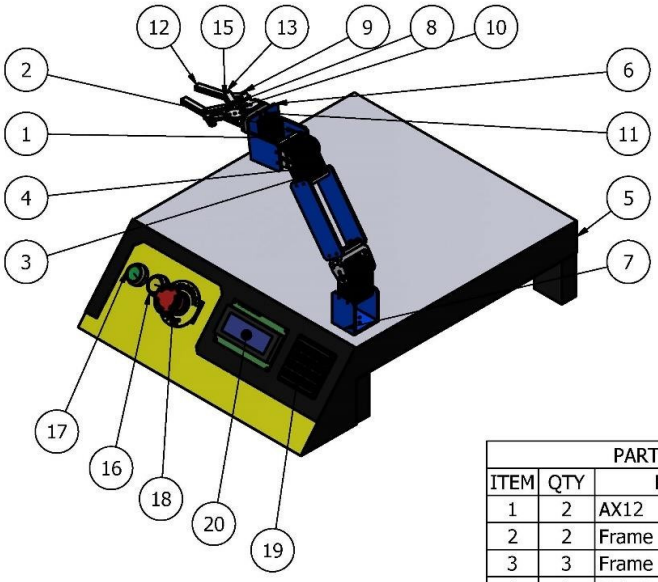
Lampiran 2 Gambar Teknik Tampak Isometri Robot Manipulator

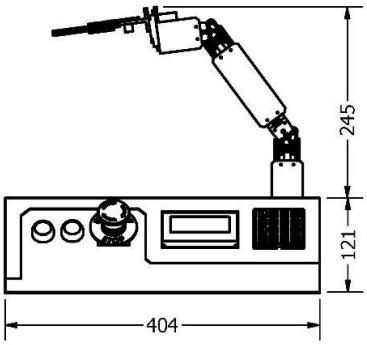


Lampiran 3 Gambar Teknik Tampak Depan Robot Manipulator



Lampiran 4 Gambar Teknik Daftar Bagian-bagian Robot Manipulator





PARTS LIST		
ITEM	QTY	PART NUMBER
1	2	AX12
2	2	Frame OF-12SH
3	3	Frame OF-12S
4	2	Link No.2
5	1	Table Robot Manipulator
6	1	Link No.3
7	1	Link No.1
8	1	Base Gripper
9	1	Arm 3
10	1	Arm 4
11	1	Towerpro MG996R
12	4	Arm 1
13	4	Arm 2
14	1	Servo SG90
15	6	Engsel Arm
16	1	Reset Button
17	1	Start Button
18	1	Emergency Button
19	1	Numpad
20	1	LCD 20x4

ROBOT MANIPULATOR				
PUTU HERDY K. & ABDULLAH GHULLAM R.	Skala 1:6	Digambar	020	002
		Diperiksa		
POLITEKNIK NEGERI UJUNG PANDANG	11 Februari 2023			

Lampiran 5 *Source Code* manip_ai.py

```
##### SECTION FOR END POINT EDIT #####
# BAGIAN INI DAPAT DIEDIT UNTUK EKSPERIMEN END POINT #

goal_pos_EP_dxl1 = 790      # Goal Position of End Point for
Dynamixel 1 (DEFAULT: 790)
goal_pos_EP_dxl2 = 270      # Goal Position of End Point for
Dynamixel 2 (DEFAULT: 270)
goal_pos_EP_dxl3 = 325      # Goal Position of End Point for
Dynamixel 3 (DEFAULT: 325)



##### BELOW IS EDIT RESTRICTION ######
##### BAGIAN DI BAWAH INI TIDAK BOLEH DIEDIT SEMBARANGAN #####

# Deep Learning libraries:
from ultralytics import YOLO
import cv2
import torch
import numpy as np

# Machine Learning libraries:
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# Dynamixel libraries:
from Ax12 import Ax12
from gpiozero import Button
from signal  import  pause
from RPLCD.i2c import CharLCD
import RPi.GPIO as GPIO
import time
import math

# - - - - - - - - - - - - - - - - -
# - - - - - - - SETUP - - - - - - -
# - - - - - - - - - - - - - - - - -
## GPIO SETUP
#START = Button(16)          # Using GPIO 16

## DYNAMIXEL SETUP
```

```
# e.g 'COM3' windows or '/dev/ttyUSB0' for Linux
Ax12.DEVICENAME = '/dev/ttyUSB0'

Ax12.BAUDRATE = 1_000_000

# sets baudrate and opens com port
Ax12.connect()

# create AX12 instance with ID 10
my_dxl_1 = Ax12(1)
my_dxl_2 = Ax12(2)
my_dxl_3 = Ax12(3)
my_dxl_1.set_moving_speed(30)
my_dxl_2.set_moving_speed(30)
my_dxl_3.set_moving_speed(30)

# - - - - - - - - - - - - - - - - -
# - - - - - - DEFINE  - - - - - -
# - - - - - - - - - - - - - - - - -
def SetAngle_4(angle):
    servo_4 = 13            # Using GPIO 13
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(servo_4, GPIO.OUT)
    pwm = GPIO.PWM(servo_4, 50)
    pwm.start(0)
    duty = angle / 18 + 2
    GPIO.output(servo_4, True)
    pwm.ChangeDutyCycle(duty)
    time.sleep(1)
    GPIO.output(servo_4, False)
    pwm.ChangeDutyCycle(0)

def SetAngle_5(angle):
    servo_5 = 12            # Using GPIO 12
    GPIO.setup(servo_5, GPIO.OUT)
    pwm = GPIO.PWM(servo_5, 50)
    pwm.start(0)
    duty = angle / 18 + 2
    GPIO.output(servo_5, True)
    pwm.ChangeDutyCycle(duty)
    time.sleep(1)
    GPIO.output(servo_5, False)
    pwm.ChangeDutyCycle(0)

def Formulasi():
```

```
    global T1, T2, T3

    L1 = 85     # mm
    L2 = 165    # mm
    L3 = 155    # mm

    # Formulasi Invers Kinematics Robot Manipulator

    theta1 = math.atan2(Y, X)    # Radian

    h = Z - L1
    c3 = (X*X + Y*Y + h*h - L2*L2 - L3*L3) / (2*L2*L3) # This is
cos theta 3
    s3a = -math.sqrt(1 - c3*c3)     # For down elbow
    s3b = math.sqrt(1 - c3*c3)      # For up elbow
    theta3 = math.atan2(s3a, c3)    # Radian

    alfa = math.atan2(math.sin(math.acos(c3)) * L3, L2 +
math.cos(math.acos(c3)) * 155)
    beta = math.atan2(h, math.sqrt(X*X + Y*Y))
    theta2 = alfa + beta

    T1a = theta1 * 180.0 / math.pi # T1a is theta1 in Degree
    T2a = theta2 * 180.0 / math.pi # T2a is theta2 in Degree
    T3a = theta3 * 180.0 / math.pi  # T3a is theta3 in Degree

    T1 = 510 + (T1a / 0.29297) # Decimal Value
    T2 = 243 + (T2a / 0.29297) # Decimal Value
    T3 = 455 + (T3a / 0.29297)  # Decimal Value

###################################
########## MAIN PROGRAM #########
###################################
# Position 1:
my_dxl_1.set_goal_position(500)
my_dxl_2.set_goal_position(230)
my_dxl_3.set_goal_position(325)
SetAngle_4(80)
SetAngle_5(180)

###################################
##### DEEP LEARNING PROCESS #####
###################################
import camcapture   # Running camcapture.py
```

```
model = YOLO('best_002.pt')  # Load model

# Predict captured image using the model
results = model(source='img0000.png',
                conf=0.55,
                save=True,
                save_txt=True,
                show=True,
                show_labels=True)  # predict on an image

for result in results:
    boxes = result.boxes  # Boxes object for bbox outputs
    masks = result.masks  # Masks object for segmentation masks
outputs
    probs = result.probs  # Class probabilities for classification
outputs

tensor = boxes.xyxy.clone().detach().requires_grad_(True)
x_img = (tensor[1,0].item() + (tensor[1,2].item() -
tensor[1,0].item()) / 2) - tensor[0,0].item()
y_img = tensor[0,3].item() - (tensor[1,1].item() +
(tensor[1,3].item() - tensor[1,1].item()) / 2)

print("x_img: %.1f" % (x_img))
print("y_img: %.1f" % (y_img))


####################################
##### MACHINE LEARNING PROCESS #####
####################################

# Store the data into a data frame
df = pd.read_csv('tb_manip.csv')

# Take corresponding data into variables
xdata_img = np.array(df['x_img'])
ydata_img = np.array(df['y_img'])
xdata_manip = np.array(df['x_manip'])
ydata_manip = np.array(df['y_manip'])

# Reshape the data for sklearn
xdata_img = xdata_img.reshape((-1, 1))
ydata_img = ydata_img.reshape((-1, 1))
xdata_manip = xdata_manip.reshape((-1, 1))
ydata_manip = ydata_manip.reshape((-1, 1))
```

```
### FOR X MODEL ###
# Create a linear regression object
model = LinearRegression()
# Fit the model
model1 = model.fit(xdata_img, xdata_manip)


### FOR Y MODEL ###
# Create a linear regression object
model = LinearRegression()
# Fit the model
model2 = model.fit(ydata_img, ydata_manip)

# Predict images values into manipulator values
x_manip_pred = model1.predict(np.array([[x_img]]))
y_manip_pred = model2.predict(np.array([[y_img]]))
x_manipulator = round(x_manip_pred[0, 0])
y_manipulator = round(y_manip_pred[0, 0])

print("x_manipulator: %d" % (x_manipulator))
print("y_manipulator: %d" % (y_manipulator))


################################
##### MANIPULATOR PROCESS #####
################################

X = x_manipulator
Y = y_manipulator
Z = 15

Formulasi()

# Position 2:
print("")
my_dxl_1.set_moving_speed(30)
my_dxl_2.set_moving_speed(30)
my_dxl_3.set_moving_speed(30)
my_dxl_1.set_goal_position(500)
my_dxl_2.set_goal_position(400)
my_dxl_3.set_goal_position(325)
time.sleep(2)
```

```
# Position 3:
print("")
my_dxl_1.set_moving_speed(30)
my_dxl_2.set_moving_speed(15)
my_dxl_3.set_moving_speed(30)

my_dxl_1.set_goal_position(int(T1))
time.sleep(2)
my_dxl_2.set_goal_position(int(T2))
time.sleep(4)
my_dxl_3.set_goal_position(int(T3))
time.sleep(2)

SetAngle_4(90)
SetAngle_5(140)

print("\nDynamixel 1 Present Position: %d" %
(my_dxl_1.get_present_position()))
print("Dynamixel 2 Present Position: %d" %
(my_dxl_2.get_present_position()))
print("Dynamixel 3 Present Position: %d" %
(my_dxl_3.get_present_position()))

# Position 4:
print("")
my_dxl_2.set_moving_speed(30)
my_dxl_2.set_goal_position(400)
time.sleep(2)

# Position 5:
print("")
my_dxl_1.set_moving_speed(30)
my_dxl_2.set_moving_speed(15)
my_dxl_3.set_moving_speed(30)
my_dxl_1.set_goal_position(goal_pos_EP_dxl1)
my_dxl_2.set_goal_position(goal_pos_EP_dxl2)
my_dxl_3.set_goal_position(goal_pos_EP_dxl3)
time.sleep(3)

SetAngle_4(90)
SetAngle_5(180)

# Position 6:
print("")
my_dxl_2.set_moving_speed(30)
```

```
my_dxl_2.set_goal_position(400)
time.sleep(2)

# Position 7:
print("")
my_dxl_1.set_moving_speed(30)
my_dxl_2.set_moving_speed(10)
my_dxl_3.set_moving_speed(30)
my_dxl_1.set_goal_position(500)
my_dxl_2.set_goal_position(280)
my_dxl_3.set_goal_position(325)
time.sleep(6)

print("\nT1 : %d" % (T1))
print("T2 : %d" % (T2))
print("T3 : %d" % (T3))


print("\nDynamixel 1 Temperature: %d °C" %
(my_dxl_1.get_temperature()))
print("Dynamixel 2 Temperature: %d °C" %
(my_dxl_2.get_temperature()))
print("Dynamixel 3 Temperature: %d °C" %
(my_dxl_3.get_temperature()))
print("")

print("\nClosing Manipulator Program")
print("")
print('Selesai\n')

# Disconnect: Disconnect Servo and Clean GPIO
my_dxl_1.set_torque_enable(0)
my_dxl_2.set_torque_enable(0)
my_dxl_3.set_torque_enable(0)
Ax12.disconnect()
GPIO.cleanup()
cv2.destroyAllWindows()
```

Lampiran 6 *Source Code* manip_manual.py

```
from Ax12 import Ax12
from gpiozero import Button
from signal  import  pause
from RPLCD.i2c import CharLCD
import RPi.GPIO as GPIO
import time
import math

# Declare Variables
a = 0
b = 0

X = 0.0
Y = 0.0
Z = 0.0

L1 = 0      # 2147483666
L2 = 0
L3 = 0

h = 0.0
theta1 = 0.0
c3 = 0.0
s3 = 0.0
s3a = 0.0
theta3 = 0.0
p1 = 0.0
p2 = 0.0
theta2a = 0.0
theta2 = 0.0

T1a = 0.0
T2a = 0.0
T3a = 0.0

T1 = 0.0
T2 = 0.0
T3 = 0.0

keypressed = ''

IptKoordinat_X = ''
IptKoordinat_Y = ''
```

```
IptKoordinat_Z = ''

# - - - - - - - - - - - - - - - - -
# - - - - - - - SETUP - - - - - - -
# - - - - - - - - - - - - - - - - -
## GPIO SETUP
START = Button(16)          # Using GPIO 16

## DYNAMIXEL SETUP
# e.g 'COM3' windows or '/dev/ttyUSB0' for Linux
Ax12.DEVICENAME = '/dev/ttyUSB0'

Ax12.BAUDRATE = 1_000_000

# sets baudrate and opens com port
Ax12.connect()

# create AX12 instance with ID 10
my_dxl_1 = Ax12(1)
my_dxl_2 = Ax12(2)
my_dxl_3 = Ax12(3)
my_dxl_1.set_moving_speed(60)
my_dxl_2.set_moving_speed(60)
my_dxl_3.set_moving_speed(60)

## LCD SETUP
lcd = CharLCD('PCF8574', 0x27)

## KEYPAD SETUP
# Define the keypad pins
MATRIX = [[1, 2, 3, "A"],
          [4, 5, 6, "B"],
          [7, 8, 9, "C"],
          ["*", 0, "#", "D"]]

ROW = [6, 19, 26, 23]
COL = [17, 27, 22, 5]

# Set the pin mode for the keypad pins
for j in range(4):
    GPIO.setup(COL[j], GPIO.OUT)
    GPIO.output(COL[j], 1)

for i in range(4):
    GPIO.setup(ROW[i], GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

```
# Function to read the keypad
def keypad():
    for j in range(4):
        GPIO.output(COL[j], 0)

        for i in range(4):
            if GPIO.input(ROW[i]) == 0:
                time.sleep(0.02)
                while GPIO.input(ROW[i]) == 0:
                    pass
                return MATRIX[i][j]

        GPIO.output(COL[j], 1)
    return None

# Function to iterate keypad()
def key():
    global keypressed
    while True:
        keypressed = keypad()
        if keypressed is not None:
            time.sleep(0.02)
            break

# - - - - - - - - - - - - - - - - -
# - - - - - - DEFINE  - - - - - -
# - - - - - - - - - - - - - - - - -
def SetAngle_4(angle):
    # Servo_4 Setup
    servo_4 = 12            # Using GPIO 12
    GPIO.setup(servo_4, GPIO.OUT)
    pwm = GPIO.PWM(servo_4, 50)
    pwm.start(0)
    duty = angle / 18 + 2
    GPIO.output(servo_4, True)
    pwm.ChangeDutyCycle(duty)
    time.sleep(1)
    GPIO.output(servo_4, False)
    pwm.ChangeDutyCycle(0)

def SetAngle_5(angle):
    # Servo_5 Setup
    servo_5 = 13            # Using GPIO 13
    GPIO.setup(servo_5, GPIO.OUT)
```

```
    pwm = GPIO.PWM(servo_5, 50)
    pwm.start(0)
    duty = angle / 18 + 2
    GPIO.output(servo_5, True)
    pwm.ChangeDutyCycle(duty)
    time.sleep(1)
    GPIO.output(servo_5, False)
    pwm.ChangeDutyCycle(0)

def InputKoordinat_X():
    global a, b, X, IptKoordinat_X, keypressed
    count = 0
    while True:
        IptKoordinat_X = ''
        X = 0
        print('LCD Homepage X')
        lcd.clear()
        lcd.cursor_pos = (0, 0)
        lcd.write_string('INPUT KOORDINAT X,Y,Z')
        lcd.cursor_pos = (1, 0)
        lcd.write_string('KOORDINAT X:')
        lcd.cursor_pos = (3, 0)
        lcd.write_string('TEKAN D JIKA SELESAI')

        while True:
            key()
            if keypressed == 'D':
                print(keypressed)
                a = 0
                lcd.clear()
                return
            else:
                count += 1
                if keypressed == 'C':
                    print('C2')
                    lcd.clear()
                    lcd.cursor_pos = (1, 0)
                    lcd.write_string(' INPUT BERHASIL ')
                    lcd.cursor_pos = (2, 0)
                    lcd.write_string('   TERHAPUS   ')
                    a = 0
                    time.sleep(1)
                    break
                elif count >= 0 and count <= 20:
                    lcd.cursor_pos = (2, count)
```

```
                    lcd.write_string(str(keypressed))
                    a += 1
                    IptKoordinat_X =
f"{IptKoordinat_X}{keypressed}"
                    X = int(IptKoordinat_X)
                    print(X)
                    if keypressed == 'C':
                        print('C1')
                        lcd.clear()
                        lcd.cursor_pos = (1, 0)
                        lcd.write_string('  INPUT BERHASIL
')
                        lcd.cursor_pos = (2, 0)
                        lcd.write_string('   TERHAPUS   ')
                        a = 0
                        time.sleep(1)
                        break
        print('C3')

def InputKoordinat_Y():
    global a, b, Y, IptKoordinat_Y, keypressed
    count = 0
    while True:
        IptKoordinat_Y = ''
        Y = 0
        print('LCD Homepage Y')
        lcd.clear()
        lcd.cursor_pos = (0, 0)
        lcd.write_string('INPUT KOORDINAT X,Y,Z')
        lcd.cursor_pos = (1, 0)
        lcd.write_string('KOORDINAT Y:')
        lcd.cursor_pos = (3, 0)
        lcd.write_string('TEKAN D JIKA SELESAI')

        while True:
            key()
            if keypressed == 'D':
                print(keypressed)
                a = 0
                lcd.clear()
                return
            else:
                count += 1
                if keypressed == 'C':
                    print('C2')
```

```
                    lcd.clear()
                    lcd.cursor_pos = (1, 0)
                    lcd.write_string(' INPUT BERHASIL ')
                    lcd.cursor_pos = (2, 0)
                    lcd.write_string('   TERHAPUS   ')
                    a = 0
                    time.sleep(1)
                    break
                elif count >= 0 and count <= 20:
                    lcd.cursor_pos = (2, count)
                    lcd.write_string(str(keypressed))
                    a += 1
                    IptKoordinat_Y =
f"{IptKoordinat_Y}{keypressed}"
                    print(IptKoordinat_Y)
                    Y = int(IptKoordinat_Y)
                    print(Y)
                    if keypressed == 'C':
                        print('C1')
                        lcd.clear()
                        lcd.cursor_pos = (1, 0)
                        lcd.write_string('  INPUT BERHASIL
')
                        lcd.cursor_pos = (2, 0)
                        lcd.write_string('   TERHAPUS   ')
                        a = 0
                        time.sleep(1)
                        break
        print('C3')
        print(Y)

def InputKoordinat_Z():
    global a, b, Z, IptKoordinat_Z, keypressed
    count = 0
    while True:
        IptKoordinat_Z = ''
        Z = 0
        print('LCD Homepage Z')
        lcd.clear()
        lcd.cursor_pos = (0, 0)
        lcd.write_string('INPUT KOORDINAT X,Y,Z')
        lcd.cursor_pos = (1, 0)
        lcd.write_string('KOORDINAT Z:')
        lcd.cursor_pos = (3, 0)
        lcd.write_string('TEKAN D JIKA SELESAI')
```

```
        while True:
            key()
            if keypressed == 'D':
                print(keypressed)
                a = 0
                lcd.clear()
                return
            else:
                count += 1
                if keypressed == 'C':
                    print('C2')
                    lcd.clear()
                    lcd.cursor_pos = (1, 0)
                    lcd.write_string(' INPUT BERHASIL ')
                    lcd.cursor_pos = (2, 0)
                    lcd.write_string('   TERHAPUS   ')
                    a = 0
                    time.sleep(1)
                    break
                elif count >= 0 and count <= 20:
                    lcd.cursor_pos = (2, count)
                    lcd.write_string(str(keypressed))
                    a += 1
                    IptKoordinat_Z =
f"{IptKoordinat_Z}{keypressed}"
                    print(IptKoordinat_Z)
                    Z = int(IptKoordinat_Z)
                    print(Z)
                    if keypressed == 'C':
                        print('C1')
                        lcd.clear()
                        lcd.cursor_pos = (1, 0)
                        lcd.write_string('  INPUT BERHASIL
')
                        lcd.cursor_pos = (2, 0)
                        lcd.write_string('   TERHAPUS   ')
                        a = 0
                        time.sleep(1)
                        break
        print('C3')
        print(Z)

def Formulasi():
    global L1, L2, L3
```

```
    global h, Z
    global X, Y, theta1
    global c3, s3, s3a, theta3
    global theta2
    global T1a, T2a, T3a
    global T1, T2, T3

    L1 = 85     # mm
    L2 = 165    # mm
    L3 = 155    # mm

    # Formulasi Invers Kinematics Robot Manipulator
    theta1 = math.atan2(Y, X)    # Radian

    h = Z - L1  # mm    # This is for theta3
    c3 = (X*X + Y*Y + h*h - L2*L2 - L3*L3) / (2*L2*L3)  #
This is cos theta 3
    s3 = -math.sqrt(1 - c3*c3) # For down elbow
    s3a = math.sqrt(1 - c3*c3) # For up elbow
    theta3 = math.atan2(s3, c3) # Radian

    alfa = math.atan2(h , math.sqrt(X*X + Y*Y))
    beta = math.atan2(math.sin(math.acos(c3)) * L3, L2 +
math.cos(math.acos(c3)) * 155)
    theta2 = alfa + beta

    T1a = theta1 * 180.0 / math.pi  # Degree    # T1a is
theta1 in Degree
    T2a = theta2 * 180.0 / math.pi  # Degree    # T2a is
theta2 in Degree
    T3a = theta3 * 180.0 / math.pi  # Degree    # T3a is
theta3 in Degree

    T1 = 510 + (T1a / 0.29297)
    T2 = 204 + (T2a / 0.29297)
    T3 = 460 + (T3a / 0.29297)

def gerakan_1():
    global T1, T2, T3
    global val1, val2

    val1 = int((511.5 - 0) * (180 - 0) / (1023 - 0) +
0)    # Result: 90
    SetAngle_4(val1)
    time.sleep(0.04)
```

```
    val2 = int((1023 - 0) * (180 - 0) / (1023 - 0) + 0)
    SetAngle_5(val2)
    time.sleep(1)

    my_dxl_1.set_goal_position(int(T1))
    time.sleep(2)
    my_dxl_2.set_goal_position(int(T2))
    time.sleep(2)
    my_dxl_3.set_goal_position(int(T3))
    time.sleep(2)

    val2 = int((682 - 0) * (180 - 0) / (1023 - 0) + 0)
    SetAngle_5(val2)
    time.sleep(1)

    time.sleep(0.015)
    b = 2



# - - - - - - - - - - - - - - - - -
# - - - -  MAIN PROGRAM   - - - -
# - - - - - - - - - - - - - - - - -
# SECTION 1: Servo Initial Position
START.wait_for_press()
print("Pressed")
my_dxl_2.set_goal_position(358)
my_dxl_3.set_goal_position(409)
my_dxl_1.set_goal_position(512)

val1 = int((511.5 - 0) * (180 - 0) / (1023 - 0) + 0)
print("Val-1: %d" % (val1))         # Result: 90 before 74
SetAngle_4(val1)
time.sleep(0.04)
val2 = int((682 - 0) * (180 - 0) / (1023 - 0) + 0)
print("Val-2: %d" % (val2))         # Result: 120 before 0
SetAngle_5(val2)
time.sleep(0.015)
b = 1

START.wait_for_press()
print("Pressed")
```

```
# SECTION 2: Input Koordinat X
InputKoordinat_X()
print(X)
print("Ipt Koordinat X: %s" % (IptKoordinat_X))

# SECTION 3: Input Koordinat Y
InputKoordinat_Y()
print(Y)
print("Ipt Koordinat Y: %s" % (IptKoordinat_Y))

# SECTION 4: Input Koordinat Z
InputKoordinat_Z()
print(Z)
print("Ipt Koordinat Z: %s" % (IptKoordinat_Z))

START.wait_for_press()
lcd.cursor_pos = (0, 0)
lcd.write_string('INPUT X= ')
lcd.cursor_pos = (0, 10)
lcd.write_string(str(X))
lcd.cursor_pos = (1, 0)
lcd.write_string('INPUT Y= ')
lcd.cursor_pos = (1, 10)
lcd.write_string(str(Y))
lcd.cursor_pos = (2, 0)
lcd.write_string('INPUT Z= ')
lcd.cursor_pos = (2, 10)
lcd.write_string(str(Z))
lcd.cursor_pos = (3, 0)
lcd.write_string("START=INPUTAN KEDUA")
time.sleep(2)
Formulasi()

print(" ")
print(type(X))
print(type(Y))
print("X : %d" % (X))
print("Y : %d" % (Y))
print("theta1 : %d" % (math.degrees(theta1)))
print(type(theta1))

print(" ")
print("h : %d" % (h))
print("c3 : %d" % (math.degrees(c3)))
print("s3 : %d" % (math.degrees(s3)))
```

```
print("s3a : %d" % (math.degrees(s3a)))
print("theta3 : %d" % (math.degrees(theta3)))
print(" ")
print("p1 : %d" % (p1))
print("p2 : %d" % (p2))
print("theta2a : %d" % (math.degrees(theta2a)))
print("theta2 : %d" % (math.degrees(theta2)))
print(" ")
print("T1a : %d" % (T1a))
print("T2a : %d" % (T2a))
print("T3a : %d" % (T3a))
print(" ")
print("T1 : %d" % (T1))
print("T2 : %d" % (T2))
print("T3 : %d" % (T3))

START.wait_for_press()
gerakan_1()
time.sleep(2)

START.wait_for_press()
print('Selesai')

# SECTION 3: Disconnect Servo and Clean GPIO
my_dxl_1.set_torque_enable(0)
my_dxl_2.set_torque_enable(0)
my_dxl_3.set_torque_enable(0)
Ax12.disconnect()
GPIO.cleanup()
```

Lampiran 7 Biodata Penulis

Putu Herdy Kurniawan, lahir di Kutai Timur, Kalimantan Timur tahun 1999. Penulis pernah belajar di George Mason University di USA, mengenyam pendidikan di Tomsk Polytechnic University dan Bauman Moscow State Technical University di Rusia lalu belajar di Fakultas *Computer Science and Information Engineering* Asia University.

Lampiran 8 Lembar Revisi

Lembar 1

LAMPIRAN BERITA ACARA PELAKSANAAN
UJIAN SIDANG SKRIPSI

Nama Mahasiswa : Putu Herdy Kurniawan

NIM : 444 19 020

Catatan/Daftar Revisi Penguji:

No.	Nama Dosen	Uraian	Tanda Tangan
1.	Prof. A.M Shiddiq	-Penulisan - Pergerakan robot. -	08/09/23
2.	Firman Hamzah	- Penggunaan matematis dan deep learning - Wiring Elektronik tambahkan ke robot	
3.	Mukhtar	- penulisan - Hal 78 - 95 dipindahkan menjadi lampiran. - Rekomendasi sistem monitor menggunakan GUI atau sejenisnya.	
4.	Abdul Kadir Muhammad.	- Machine learning dan deep learning - Ada Arsitektur lanjutan..	